Proyecto Semestral

Integrantes:

- Tomás Alfaro
- Francisca Alvarez
- Matias Umaña
- Tomás Pantoja

1. Caso escrito que describa con todo detalle el problema, la necesidad y los requisitos lógicos a considerar.

Los estudiantes de la universidad Adolfo Ibáñez se han visto afectados de manera significativa luego de la pandemia ocasionada por el Covid-19. El año 2022, luego de su formato virtual durante dos años se llegó a la decisión de un formato 100% presencial, en esta etapa comenzaron a haber contagios para los estudiantes, tanto en la universidad como fuera de ella, por lo que hubo cuarentenas individuales para las personas contagiadas. La gestión de las justificaciones ha funcionado de una manera poco óptima y amigable para el equipo de secretaría y la seguridad de los alumnos, ya que al entrar las justificaciones por solo una vía, el trabajo de secretaría pasa a ser menos eficiente porque tienen que estar evaluando distintos casos de justificativos diariamente. Por el mismo motivo, el equipo de secretaría se ve imposibilitado a mandar un correo a todos los profesores y alumnos relacionados al justificante, informándoles el nuevo caso de Covid-19. Esto serviría para que estas personas estén atentas a la aparición de síntomas. Es por ello que se comenzara a atacar el funcionamiento de la web de la UAI de forma que haya una sección de solo contagios por Covid-19, con esto se logrará mayor eficiencia para una respuesta de la debida justificación y habrá una mayor facilidad para la comunicación del nuevo caso de contagio por Covid-19. Además, los profesores al ser comunicados, ya sabrán que el alumno faltará en su periodo de cuarentena, para justificarlo de pruebas y controles si es que se necesitara esto como justificación de asistencia a las materias de la semana.

Requisitos lógicos a considerar:

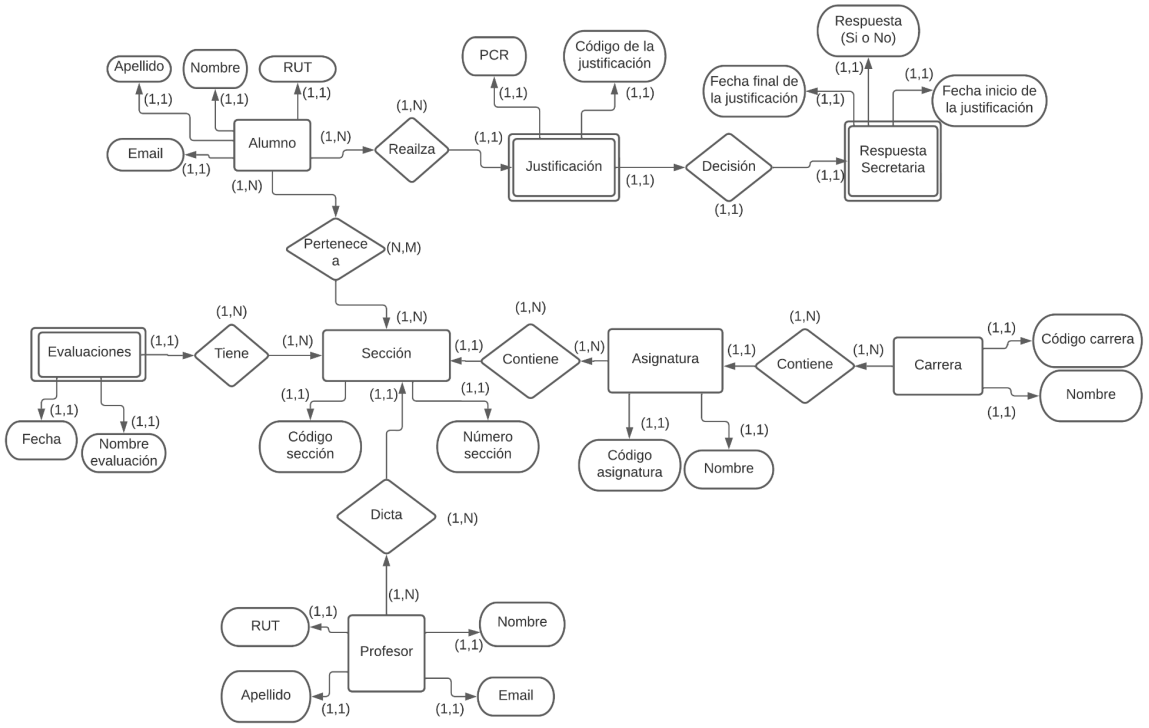
El formato del resultado PCR será siempre URL, todo mail enviado al alumno (mail con el resultado PCR) será en PDF pero tendrá un link con el URL del diagnóstico, para poder llenar así la justificación que será enviada a secretaría posteriormente.

2. Modelo Entidad Relación con, al menos, 6 entidades fuertes y débiles en total.

Una entidad fuerte será representada por un rectángulo y en su interior se encontrará el nombre de dicha entidad. Las entidades débiles serán representadas con un rectángulo doble, uno dentro de otro. Además, cabe decir que los atributos serán modelados con un óvalo, dentro de estos estará el nombre del atributo, y se unirán con la entidad correspondiente mediante una línea identificando su cardinalidad. Por último, cada entidad tendrá una inter-relación con otra, esta inter-relación será representada por un rombo y definirá la asociación de las entidades involucradas.

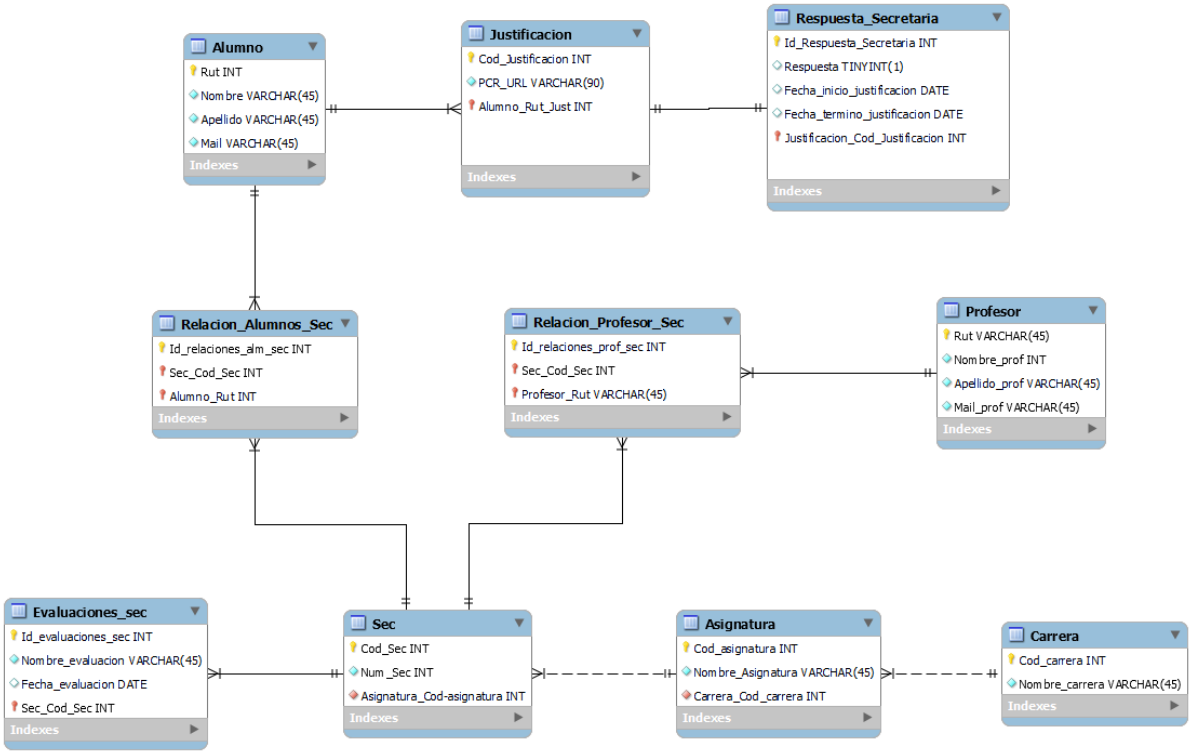
Las entidades que serán trabajadas serán 8 y se mostraran a continuación. Dentro de las 8 entidades, 3 son entidades débiles, entidades que dependen de otras para su existencia por lo que 5 son entidades fuertes, las cuales no dependen de otras para existir.

- Alumno: Rut, Nombre, Apellido, Mail.
- Justificación: Cod-Justificación, PCR.
- Respuesta-Secretaria: Respuesta, Fecha-inicio-justificacion, Fecha-Término-justificación.
- Profesor: Rut, Nombre_prof, Apellido_prof, Mail_prof.
- Evaluaciones-Asignaturas: Curso_Cod, Nombre-evaluación, Fecha-evaluación.
- Sec: Cod_Sec, Num_Sec.
- Asignatura: Cod-asignatura, Nombre_Asignatura.
- Carrera: Cod_carrera, Nombre_carrera.1



3. Modelo relacional

Luego de dar a conocer lo que es el modelo MER se mostrará a continuación un modelo más representativo de lo que se contiene en las tablas de datos. Entonces, el modelo relacional será el modelo más ilustrativo y funcional en la base de datos, ya que cada atributo estará dentro del rectángulo, dando a conocer una serie de datos en ella, cada atributo de la entidad tendrá una columna de datos correspondiente y ordenados mediante su PRIMARY KEY.



4. Modelo físico en SQL validado en tercera forma normal

A continuación se dará a conocer el modelo físico del detalle del problema en la UAI, el cual viene describiendo cómo se va a implementar una base de datos sobre un DBMS específico.

Para una correcta implementación de los datos y posterior trabajo con estos mismos se es necesario realizar una normalización. La normalización consiste en asegurar que cada tabla trate de un solo concepto y no tenga en esta misma datos repetitivos. Además, la normalización consiste en llevar a cabo una cantidad finita de reglas a las relaciones que se dieron a conocer en el proceso de MER a modelo relacional. Por último, se dejan de lado las inconsistencias, asegurándose que la distribución de los atributos entre las diferentes entidades sean las correctas. Para llevar a cabo lo anterior se debe llegar a 3ra forma normal, dicho en otras palabras, no debe haber más de un atributo en un campo, todos los campos deben depender de la llave primaria de la tabla y se debe eliminar cualquier dependencia transitiva que se encuentre.

```
-- MySQL Workbench Forward Engineering

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,ERROR
_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

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

-- Schema Justificaciones_Covid_UAI

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

CREATE SCHEMA IF NOT EXISTS `Justificaciones_Covid_UAI` DEFAULT CHARACTER SET utf8 ;

USE `Justificaciones_Covid_UAI` ;

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

-- Table `Justificaciones_Covid_UAI`.`Alumno`

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

CREATE TABLE IF NOT EXISTS `Justificaciones_Covid_UAI`.`Alumno` (

  `Rut` INT NOT NULL,

  `Nombre` VARCHAR(45) NOT NULL,

  `Apellido` VARCHAR(45) NOT NULL,

  `Mail` VARCHAR(45) NOT NULL,
```

```
  PRIMARY KEY (`Rut`),
  UNIQUE INDEX `Mail_UNIQUE` (`Mail` ASC) ,
  UNIQUE INDEX `Rut_UNIQUE` (`Rut` ASC) )
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `Justificaciones_Covid_UAI`.`Carrera`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Justificaciones_Covid_UAI`.`Carrera` (
  `Cod_carrera` INT NOT NULL AUTO_INCREMENT,
  `Nombre_carrera` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Cod_carrera`),
  UNIQUE INDEX `Nombre_carrera_UNIQUE` (`Nombre_carrera` ASC) ,
  UNIQUE INDEX `Cod_carrera_UNIQUE` (`Cod_carrera` ASC) )
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `Justificaciones_Covid_UAI`.`Asignatura`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Justificaciones_Covid_UAI`.`Asignatura` (
  `Cod_asignatura` INT NOT NULL AUTO_INCREMENT,
  `Nombre_Asignatura` VARCHAR(45) NOT NULL,
  `Carrera_Cod_carrera` INT NOT NULL,
  PRIMARY KEY (`Cod_asignatura`),
  INDEX `fk_Asignatura_Carrera1_idx` (`Carrera_Cod_carrera` ASC) ,
  UNIQUE INDEX `Nombre_Asignatura_UNIQUE` (`Nombre_Asignatura` ASC) ,
  UNIQUE INDEX `Cod_asignatura_UNIQUE` (`Cod_asignatura` ASC) ,
  CONSTRAINT `fk_Asignatura_Carrera1`
    FOREIGN KEY (`Carrera_Cod_carrera`)
    REFERENCES `Justificaciones_Covid_UAI`.`Carrera` (`Cod_carrera`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `Justificaciones_Covid_UAI`.`Sec`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Justificaciones_Covid_UAI`.`Sec` (
  `Cod_Sec` INT NOT NULL AUTO_INCREMENT,
  `Num_Sec` INT NOT NULL,
  `Asignatura_Cod-asignatura` INT NOT NULL,
  PRIMARY KEY (`Cod_Sec`),
  INDEX `fk_Curso_Asignatura1_idx` (`Asignatura_Cod-asignatura` ASC) ,
  UNIQUE INDEX `Cod_Sec_UNIQUE` (`Cod_Sec` ASC) ,
  CONSTRAINT `fk_Curso_Asignatura1`
    FOREIGN KEY (`Asignatura_Cod-asignatura`)
    REFERENCES `Justificaciones_Covid_UAI`.`Asignatura` (`Cod_asignatura`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `Justificaciones_Covid_UAI`.`Relacion_Alumnos_Sec`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Justificaciones_Covid_UAI`.`Relacion_Alumnos_Sec` (
  `Id_relaciones_alm_sec` INT NOT NULL AUTO_INCREMENT,
  `Sec_Cod_Sec` INT NOT NULL,
  `Alumno_Rut` INT NOT NULL,
  PRIMARY KEY (`Id_relaciones_alm_sec`, `Sec_Cod_Sec`, `Alumno_Rut`),
  INDEX `fk_Relacion_Alumnos_Sec_Sec1_idx` (`Sec_Cod_Sec` ASC) ,
  INDEX `fk_Relacion_Alumnos_Sec_Alumno1_idx` (`Alumno_Rut` ASC) ,
  UNIQUE INDEX `Id_relaciones_alm_sec_UNIQUE` (`Id_relaciones_alm_sec` ASC) ,
  CONSTRAINT `fk_Relacion_Alumnos_Sec_Sec1`
    FOREIGN KEY (`Sec_Cod_Sec`)
    REFERENCES `Justificaciones_Covid_UAI`.`Sec` (`Cod_Sec`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Relacion_Alumnos_Sec_Alumno1`
    FOREIGN KEY (`Alumno_Rut`)
    REFERENCES `Justificaciones_Covid_UAI`.`Alumno` (`Rut`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Justificaciones_Covid_UAI`.`Profesor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Justificaciones_Covid_UAI`.`Profesor` (
  `Rut` VARCHAR(45) NOT NULL,
  `Nombre_prof` INT NOT NULL,
  `Apellido_prof` VARCHAR(45) NOT NULL,
  `Mail_prof` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Rut`),
  UNIQUE INDEX `Mail_prof_UNIQUE` (`Mail_prof` ASC) ,
  UNIQUE INDEX `Rut_UNIQUE` (`Rut` ASC) )
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Justificaciones_Covid_UAI`.`Relacion_Profesor_Sec`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Justificaciones_Covid_UAI`.`Relacion_Profesor_Sec` (
  `Id_relaciones_prof_sec` INT NOT NULL AUTO_INCREMENT,
  `Sec_Cod_Sec` INT NOT NULL,
  `Profesor_Rut` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Id_relaciones_prof_sec`, `Sec_Cod_Sec`, `Profesor_Rut`),
  INDEX `fk_Relacion_Profesor_Sec_Sec1_idx` (`Sec_Cod_Sec` ASC) ,
  INDEX `fk_Relacion_Profesor_Sec_Profesor1_idx` (`Profesor_Rut` ASC) ,
  UNIQUE INDEX `Id_relaciones_prof_sec_UNIQUE` (`Id_relaciones_prof_sec` ASC) ,
  CONSTRAINT `fk_Relacion_Profesor_Sec_Sec1`
```

```
    FOREIGN KEY (`Sec_Cod_Sec`)
    REFERENCES `Justificaciones_Covid_UAI`.`Sec` (`Cod_Sec`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Relacion_Profesor_Sec_Profesor1`
    FOREIGN KEY (`Profesor_Rut`)
    REFERENCES `Justificaciones_Covid_UAI`.`Profesor` (`Rut`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `Justificaciones_Covid_UAI`.`Evaluaciones_sec`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Justificaciones_Covid_UAI`.`Evaluaciones_sec` (
  `Id_evaluaciones_sec` INT NOT NULL AUTO_INCREMENT,
  `Nombre_evaluacion` VARCHAR(45) NOT NULL,
  `Fecha_evaluacion` DATE NULL,
  `Sec_Cod_Sec` INT NOT NULL,
  PRIMARY KEY (`Id_evaluaciones_sec`, `Sec_Cod_Sec`),
  INDEX `fk_Evaluaciones_sec_Sec1_idx` (`Sec_Cod_Sec` ASC) ,
  UNIQUE INDEX `Id_evaluaciones_sec_UNIQUE` (`Id_evaluaciones_sec` ASC) ,
  CONSTRAINT `fk_Evaluaciones_sec_Sec1`
    FOREIGN KEY (`Sec_Cod_Sec`)
    REFERENCES `Justificaciones_Covid_UAI`.`Sec` (`Cod_Sec`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `Justificaciones_Covid_UAI`.`Justificacion`
```

```
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Justificaciones_Covid_UAI`.`Justificacion` (
  `Cod_Justificacion` INT NOT NULL AUTO_INCREMENT,
  `PCR_URL` VARCHAR(90) NOT NULL,
  `Alumno_Rut_Just` INT NOT NULL,
  PRIMARY KEY (`Cod_Justificacion`, `Alumno_Rut_Just`),
  UNIQUE INDEX `PCR_URL_UNIQUE` (`PCR_URL` ASC) ,
  UNIQUE INDEX `Cod_Justificacion_UNIQUE` (`Cod_Justificacion` ASC) ,
  INDEX `fk_Justificacion_Alumno1_idx` (`Alumno_Rut_Just` ASC) ,
  CONSTRAINT `fk_Justificacion_Alumno1`
    FOREIGN KEY (`Alumno_Rut_Just`)
    REFERENCES `Justificaciones_Covid_UAI`.`Alumno` (`Rut`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Justificaciones_Covid_UAI`.`Respuesta_Secretaria`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Justificaciones_Covid_UAI`.`Respuesta_Secretaria` (
  `Id_Respuesta_Secretaria` INT NOT NULL AUTO_INCREMENT,
  `Respuesta` TINYINT(1) NULL,
  `Fecha_inicio_justificacion` DATE NULL,
  `Fecha_termino_justificacion` DATE NULL,
  `Justificacion_Cod_Justificacion` INT NOT NULL,
  PRIMARY KEY (`Id_Respuesta_Secretaria`, `Justificacion_Cod_Justificacion`),
  UNIQUE INDEX `Id_Respuesta_Secretaria_UNIQUE` (`Id_Respuesta_Secretaria` ASC) ,
  INDEX `fk_Respuesta_Secretaria_Justificacion1_idx` (`Justificacion_Cod_Justificacion` ASC) ,
  CONSTRAINT `fk_Respuesta_Secretaria_Justificacion1`
    FOREIGN KEY (`Justificacion_Cod_Justificacion`)
    REFERENCES `Justificaciones_Covid_UAI`.`Justificacion` (`Cod_Justificacion`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION)

ENGINE = InnoDB;



SET SQL_MODE=@OLD_SQL_MODE;

SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;

SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

5. Formulario en form.io que permita poblar las distintas tablas con datos

Tabla sección: https://pro.formview.io/#/bescakozwcbhbvp/seccion?header=1&reset=1

Tabla alumno: https://pro.formview.io/#/bescakozwcbhbvp/alumnos?header=1&reset=1

Tabla profesor: https://pro.formview.io/#/bescakozwcbhbvp/profesor?header=1&reset=1

Tabla asignatura: https://pro.formview.io/#/bescakozwcbhbvp/asignatura?header=1&reset=1

Tabla justificación:
https://pro.formview.io/#/bescakozwcbhbvp/justificacion?header=1&reset=1

Tabla carrera: https://pro.formview.io/#/bescakozwcbhbvp/carrera?header=1&reset=1

Tabla evaluaciones:
https://pro.formview.io/#/bescakozwcbhbvp/evaluaciones?header=1&reset=1

Tabla respuesta secretaria:
https://pro.formview.io/#/bescakozwcbhbvp/respuestasecretaria?header=1&reset=1

Tabla relación profesor sección:

https://pro.formview.io/#/tyqdmdbrjlzgcze/relacionprofesorseccion?header=1&reset=1

Tabla relación alumnos sección:

https://pro.formview.io/#/tyqdmdbrjlzgcze/relacionalumnosseccion?header=1&reset=1

6. Script de python que rescate la información desde form.io y la cargue en la base de datos sql

Los códigos python van adjuntos en la carpeta “CodigosPython”.

7. 12 consultas SQL basadas en el modelo que permitan contestar preguntas relevantes del caso. Considere los siguientes tipos de consultas:

a. 4 consultas simples a 4 tablas específicas.
b. 4 consultas que contengan JOIN entre tablas.
c. 4 consultas que agrupen información, contando registros, sumando valores de algún campo específico, sacando promedios de valores de algún campo específico.

1. Ordenar las asignaturas con respecto a sus carreras

   ```
   SELECT * FROM carrera, asignatura
   WHERE cod_carrera = carrera_cod_carrera
   ```

2. Visualizar todos los alumnos

   ```
   Select * from alumno
   ```

3. Cantidad de justificación actuales

   ```
   SELECT COUNT(Cod_Justificacion) FROM Justificacion
   ```

4. Mostrar las asignaturas disponibles

   ```
   select nombre_asignatura from asignatura
   ```

5. Cantidad de secciones por asignatura “x”

   ```
   SELECT * FROM Sec INNER JOIN Asignatura ON Sec.Asignatura_Cod-asignatura
   = Asignatura.Cod_asignatura
   SELECT COUNT(Cod_Sec) FROM Sec WHERE Nombre_Asignatura = “ ”
   ```

6. Obtener el mail de todos los alumnos relacionados al alumno “x”

   ```
   SELECT
   mail,
   a1.alumno_rut Alumno_contagiado,
   a2.alumno_rut Alumno_relacionado
   FROM (relacion_alumnos_sec  a1
   INNER JOIN relacion_alumnos_sec a2
   ON a1.alumno_rut != a2.alumno_rut)
   LEFT JOIN alumno on a2.alumno_rut = rut
   and a1.sec_cod_sec = a2.sec_cod_sec
   WHERE a1.alumno_rut = 19890244  ;
   ```

7. Tomar los datos, justificaciones pendientes (datos nulos) y visualizar cuántas son

   SELECT * FROM Respuesta_Secretaria WHERE Respuesta IS NULL

8. Mostrar cuántos alumnos han justificado hasta el momento

   SELECT COUNT(RUT) FROM Alumno INNER JOIN Justificacion ON Alumno.Rut = Justificacion.Alumno_Rut_Just

9. Mostrar la cantidad de justificaciones por alumno (Rut)

   SELECT Alumno_Rut_Just, COUNT(Cod_Justificacion)
   FROM Justificacion
   GROUP BY Alumno_Rut_Just

10. Mostrar cuantas secciones imparte el profesor (Rut)

    SELECT Profesor_Rut, COUNT(Sec_Cod_Sec)
    FROM Relacion_Profesor_Sec
    GROUP BY Profesor_Rut

11. Obtener la media de las fechas de las evaluaciones de cada sección

    SELECT Sec_Cod_Sec, AVG(Fecha_evaluacion)
    FROM Evaluaciones_Sec
    GROUP BY Sec_Cod_Sec

12. Mostrar cuantas justificaciones ya fueron respondidas por la secretaria

    SELECT Justificacion_Cod_Justificacion, COUNT(Respuesta)
    FROM Respuesta_Secretaria
    GROUP BY Justificacion_Cod_Justificacion