Refactoring:
Using COUNT(column) instead of COUNT(*)

Implementación con 2 Visitors: (Para el análisis de pre/post condiciones y el refactor en sí)

Objetivo : Transformar querys con la cláusula Count(*), intercambiando "*" con el nombre o alias de una columna particular.

- ➢ Precondiciones:
  - La consulta debe ser válida.
  - La query debe contener la cláusula COUNT(*).
  - Asumimos el nombre a reemplazar por "*" existente y válida.

    (Es responsabilidad del programador proveer una PK válida)

- ➢ Postcondiciones:
  - La consulta debe seguir siendo válida.
  - No debe existir más la cláusula count(*), dada la transformación.

Pre-condiciones

Válida: “SELECT COUNT(*) FROM ALUMNOS WHERE EDAD > 18;”

Post-refactor:

```
SELECT COUNT(dni) FROM ALUMNOS WHERE EDAD > 18;
```

Inválida: SELECT * FROM ALUMNOS WHERE ID > 1;

Post-condiciones

Válida: "SELECT COUNT(id) FROM table_1;"

Post-refactor:

```
SELECT COUNT(id) FROM table_1;
```

Inválida: SELECT COUNT(*) FROM table_1;

- **Implementación**
    - CountFinderVisitor
        - Reimplementa visitExpr(`SQLiteParser.ExprContext` ctx) y actúa cómo verificador de Pre y Post Condiciones
    - CountVisitor
        - Reimplementa visitExpr() y visitParse() y genera el intercambio del símbolo “*”
    - CountRefactoring
        - Subclase de Refactoring que aprovecha patrón Template Method
    - ➔ Por último incluimos la clase `HavingAnalizerForVisitors.java` con el fin de verificar que la operación COUNT(*) esté asociada efectivamente a la cláusula SELECT y no HAVING (aprovechando los tokens de comparación).

- **Conclusiones**
    - Se utiliza un Visitor para contemplar tanto refactoring válido cómo exitoso
    - La gramática no posee token definido para la palabra clave “Count”
    - Se utiliza una v.i. de tipo: “TokenStreamRewriter” para acceder y reemplazar el símbolo STAR por una palabra elegida (Dentro de CountVisitor)