Here's a breakdown of the SQL operations and steps being performed in the provided SQL file:-

1. **Initial Data Selection**

```
SELECT * FROM world_layoffs.layoffs;
```

Step: Retrieve all the data from the `layoffs` table.

Purpose: To view or check the existing data in the `world_layoffs.layoffs` table.

2. Staging Table Creation

```
CREATE TABLE layoffs_stagging LIKE layoffs;
SELECT * FROM layoffs_stagging;
```

Step: Create a new table `layoffs_stagging` with the same structure as `layoffs`.

Purpose: Prepare a staging table to manipulate data without affecting the original dataset.

3. Data Insertion

```
INSERT layoffs_stagging
SELECT * FROM layoffs;
```

Step: Copy all the data from `layoffs` to `layoffs_stagging`.

Purpose: To have a working copy of the data in the staging table.

4. Add Row Number to Identify Duplicates

```
SELECT *, ROW_NUMBER() OVER (
 PARTITION BY
COMPANY,LOCATION,INDUSTRY,TOTAL_LAID_OFF,PERCENTAGE_LAID_OFF,`DATE`,STAGE,CO
UNTRY,FUNDS_RAISED_MILLIONS) AS ROW_NUM
FROM layoffs_stagging;
```

Step: Add a row number partitioned by specific columns.

Purpose: To identify duplicate rows within the specified columns.

5. Identifying Duplicate Rows

```
WITH duplicate_cte AS (
 SELECT *, ROW_NUMBER() OVER (
  PARTITION BY
COMPANY,LOCATION,INDUSTRY,TOTAL_LAID_OFF,PERCENTAGE_LAID_OFF,`DATE`,STAGE,CO
UNTRY,FUNDS_RAISED_MILLIONS) AS ROW_NUM
 FROM layoffs_stagging
)
SELECT *
FROM duplicate_cte
WHERE ROW_NUM > 1;
```

Step: Identify duplicates based on the `ROW_NUM` greater than 1.

Purpose: To find duplicate entries for removal or review.

6. Deleting Duplicate Rows

```
DELETE FROM duplicate_cte
WHERE ROW_NUM > 1;
```

Step: Remove duplicate rows from the `layoffs_stagging` table.

Purpose: Ensure the data contains no duplicate entries.

7. Add and Drop `ROW_NUM` Column

```
ALTER TABLE layoffs_stagging ADD COLUMN `ROW_NUM` int;
ALTER TABLE layoffs_stagging DROP COLUMN `ROW_NUM` INT;
```

Step: Add and drop the `ROW_NUM` column.

Purpose: This appears to be redundant as `ROW_NUM` is dynamically generated by the window function. It might be used temporarily for some internal process.

8. Create New Table with Updated Schema

```
CREATE TABLE layoffs_stagging2 ( ... );
```

- Step: Create a new table `layoffs_stagging2` with specific data types and columns.

- Purpose: Likely for further processing or to accommodate additional modifications with a clean schema.

9. Reinsert Data with Row Number

```
INSERT INTO layoffs_stagging2

SELECT *, ROW_NUMBER() OVER (

 PARTITION BY
COMPANY,LOCATION,INDUSTRY,TOTAL_LAID_OFF,PERCENTAGE_LAID_OFF,`DATE` ,STAGE,CO
UNTRY,FUNDS_RAISED_MILLIONS) AS ROW_NUM

FROM layoffs_stagging;
```

Step: Reinsert data into `layoffs_stagging2` with `ROW_NUM` to identify duplicates again.

Purpose: Begin working with a fresh dataset, including row numbering.

10. Delete Duplicate Rows in New Table

```
DELETE FROM layoffs_stagging2

WHERE ROW_NUM > 1;
```

Step: Remove duplicate rows from `layoffs_stagging2`.

Purpose: Clean the data in this new table by removing duplicates.

11. Standardizing Data (Company Name and Industry)

```
UPDATE layoffs_stagging2 SET company= TRIM(company);

UPDATE layoffs_stagging2 SET industry= 'Crypto' WHERE industry LIKE 'crypto%';

UPDATE layoffs_stagging2 SET country= 'United States' WHERE country LIKE 'United States.%';
```

Step: Trim whitespaces from `company`, standardize `industry` names, and update `country` values.

Purpose: Clean and standardize textual data to ensure consistency across records.

12. Modify Column Data Type

```
ALTER TABLE layoffs_stagging2 MODIFY COLUMN `date` DATE;
```

Step: Change the data type of the `date` column.

Purpose: Convert the `date` column to a `DATE` format for accurate date-based operations.

13. Handle Missing or Null Values

```
SELECT * FROM layoffs_stagging2 WHERE total_laid_off IS NULL AND percentage_laid_off IS NULL;

DELETE FROM layoffs_stagging2 WHERE total_laid_off IS NULL AND percentage_laid_off IS NULL;
```

Step: Identify and delete rows where `total_laid_off` and `percentage_laid_off` are both null.

Purpose: Remove incomplete or irrelevant records from the dataset.

14. Fill Missing Industry Values

```
UPDATE layoffs_stagging2 t1
JOIN layoffs_stagging2 t2 ON t1.company=t2.company
SET t1.industry = t2.industry
WHERE t1.industry IS NULL AND t2.industry IS NOT NULL;
```

Step: Update null `industry` fields based on matching records in the same company and location.

Purpose: Impute missing `industry` values using available data from other rows.

15. Final Cleanup

```
ALTER TABLE layoffs_stagging2 DROP COLUMN ROW_NUM;
```

Step: Drop the `ROW_NUM` column.

Purpose: Clean up unnecessary columns after data processing.

This set of operations performs data cleaning, standardization, duplicate removal, and imputation on a dataset related to company layoffs.