```sql
-- Use the target database
USE world_layoffs;

-- Preview the original dataset
SELECT *
FROM world_layoffs.layoffs;

-------------------------------------------------------------------------------
-- 1. Remove Duplicates
-------------------------------------------------------------------------------

-- Create a staging table with the same structure as the original
CREATE TABLE layoffs_staging LIKE layoffs;

-- Confirm staging table creation
SELECT * FROM layoffs_staging;

-- Load all data into the staging table
INSERT INTO layoffs_staging
SELECT * FROM layoffs;

-- Assign a row number to each record partitioned by key fields to identify duplicates
SELECT
    *,
    ROW_NUMBER() OVER (
        PARTITION BY company, location, industry, total_laid_off, percentage_laid_off, `date`
        ORDER BY company
    ) AS row_num
FROM layoffs_staging;

-- Use a CTE to filter out duplicate records (row_num > 1)
WITH duplicate_cte AS (
    SELECT
        *,
        ROW_NUMBER() OVER (
            PARTITION BY company, location, industry, total_laid_off, percentage_laid_off, `date`,
                         stage, country, funds_raised_millions
            ORDER BY company
        ) AS row_num
    FROM layoffs_staging
)
SELECT *
FROM duplicate_cte
WHERE row_num > 1;   -- Show true duplicates
```

```
-- Example check for a specific company
SELECT *
FROM layoffs_staging
WHERE company = 'Casper';

-- Create a second staging table with an extra row_num column to hold deduplication results
CREATE TABLE layoffs_staging2 (
    company                 TEXT,
    location                TEXT,
    industry                TEXT,
    total_laid_off          INT DEFAULT NULL,
    percentage_laid_off     TEXT,
    `date`                  TEXT,
    stage                   TEXT,
    country                 TEXT,
    funds_raised_millions   INT DEFAULT NULL,
    row_num                 INT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-- Confirm structure of second staging table
SELECT * FROM layoffs_staging2;

-- Populate second staging table and calculate row numbers for deduplication
INSERT INTO layoffs_staging2
SELECT
    company,
    location,
    industry,
    total_laid_off,
    percentage_laid_off,
    `date`,
    stage,
    country,
    funds_raised_millions,
    ROW_NUMBER() OVER (
        PARTITION BY company, location, industry, total_laid_off, percentage_laid_off, `date`,
                stage, country, funds_raised_millions
        ORDER BY company
    ) AS row_num
FROM layoffs_staging;

-- Delete duplicate rows, keeping only the first occurrence
DELETE FROM layoffs_staging2
```

```
WHERE row_num > 1;

-- Validate deduplication result
SELECT * FROM layoffs_staging2;

--------------------------------------------------------------------------------
-- 2. Standardize the Data
--------------------------------------------------------------------------------

-- Trim whitespace in the company column
UPDATE layoffs_staging2
SET company = TRIM(company);

-- Inspect distinct industries to find inconsistent values
SELECT DISTINCT industry
FROM layoffs_staging2;

-- Preview rows where industry starts with 'Crypto'
SELECT *
FROM layoffs_staging2
WHERE industry LIKE 'Crypto%';

-- Standardize all 'Crypto...' values to 'Crypto'
UPDATE layoffs_staging2
SET industry = 'Crypto'
WHERE industry LIKE 'Crypto%';

-- Inspect distinct country values to find trailing periods or variations
SELECT DISTINCT country
FROM layoffs_staging2
ORDER BY country;

-- Preview 'United States' variations
SELECT *
FROM layoffs_staging2
WHERE country LIKE 'United States%'
ORDER BY country;

-- Trim trailing periods from country names
UPDATE layoffs_staging2
SET country = TRIM(TRAILING '.' FROM country)
WHERE country LIKE 'United States%';

-- Convert date strings to proper DATE format
```

```
-- First, parse the text dates
SELECT
    `date`,
    STR_TO_DATE(`date`, '%m/%d/%Y') AS parsed_date
FROM layoffs_staging2;

-- Update the date column with parsed DATE values
UPDATE layoffs_staging2
SET `date` = STR_TO_DATE(`date`, '%m/%d/%Y');

-- Alter the column type from TEXT to DATE now that data is standardized
ALTER TABLE layoffs_staging2
MODIFY COLUMN `date` DATE;

-------------------------------------------------------------------------------
-- 3. Handle Null or Blank Values
-------------------------------------------------------------------------------

-- Identify rows missing both total_laid_off and percentage_laid_off
SELECT *
FROM layoffs_staging2
WHERE total_laid_off IS NULL
  AND percentage_laid_off IS NULL;

-- Convert empty strings in industry to NULL for consistency
UPDATE layoffs_staging2
SET industry = NULL
WHERE industry = '';

-- Verify industry NULLs or blanks
SELECT *
FROM layoffs_staging2
WHERE industry IS NULL
   OR industry = '';

-- Example check for specific company rows
SELECT *
FROM layoffs_staging2
WHERE company = 'Airbnb';

-- Use self-join to fill missing industry values based on other rows for the same company &
location
UPDATE t1
JOIN layoffs_staging2 AS t2
```

```
  ON t1.company = t2.company
 AND t1.location = t2.location
SET t1.industry = t2.industry
WHERE (t1.industry IS NULL OR t1.industry = '')
  AND t2.industry IS NOT NULL;

-------------------------------------------------------------------------------
-- 4. Remove Unnecessary Columns and Final Cleanup
-------------------------------------------------------------------------------

-- Confirm rows still missing both key metrics before deletion
SELECT *
FROM layoffs_staging2
WHERE total_laid_off IS NULL
  AND percentage_laid_off IS NULL;

-- Delete rows missing both total_laid_off and percentage_laid_off
DELETE FROM layoffs_staging2
WHERE total_laid_off IS NULL
  AND percentage_laid_off IS NULL;

-- Preview final cleaned dataset
SELECT * FROM layoffs_staging2;

-- Drop the temporary row_num column now that deduplication is complete
ALTER TABLE layoffs_staging2
DROP COLUMN row_num;

-------------------------------------------------------------------------------
-- 5. Export Cleaned Data to CSV
-------------------------------------------------------------------------------

-- Export the final cleaned table to the secure-file-priv directory
SELECT *
FROM layoffs_staging2
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/layoffs_cleaned.csv'
CHARACTER SET utf8mb4
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\r\n';
```