PRJ 2 LA Crime Data Refresh Documentation:

Data Details:

Data Source:
https://catalog.data.gov/dataset/crime-data-from-2020-to-present

Data Dictionary:
https://data.lacity.org/Public-Safety/Crime-Data-from-2020-to-Present/2nrs-mtv8/about_data

SPECIAL NOTE:
The following documentation will be detailing how new codes purpose and outcomes. Due to the similarities between PRJ1 and PRJ2's code this document will reference the alterations to the original code and the new code added. The altered code will not be discussed in as much detail as the newly added code. Review the PRJ1 documentation if not done so already before reading this documentation for the best grasp of the code.

Slight Alterations

1. Where tables were created in the previous code, temp tables are now created in their place and the table names are the same except for the addition of **_temp** to the end.
2. Constraints, primary keys, and foreign keys have not been included when creating temp codes. These will be applied when loading the data into the actual tables.

Major Alterations and Additions:

Filtering Out Redundant Data Rows

```
-- Filtering data.
BEGIN TRANSACTION;

-- creating filtered  data to load into la_crimes_temp.
CREATE TEMP TABLE filtered_data AS
   Select * FROM
   (SELECT * FROM la_crimes_temp
   EXCEPT
   SELECT * FROM la_crimes)
   ORDER BY date_occ;

-- Dropping all rows to make room for filtered data.
```

```
TRUNCATE TABLE la_crimes_temp;

-- Load in the filtered data.
INSERT INTO la_crimes_temp
SELECT DISTINCT * FROM filtered_data;

-- Filtered table is now waste. Drop this table.
DROP TABLE filtered_data;

COMMIT;
```

- The data from the new data set is taken and filtered using the except set operator to exclude all data found in the previously current data set in the form of a subquery.
- It is loaded into the filter table and then after truncating the crimes_data_temp table it is then loaded into the table and then dropped to remove waste from the session.

Improved Approach to Removing Excess Spaces

```
--- Clean Address fields of abnormal spaces.
DO $$
DECLARE
   i INT;
BEGIN
   FOR i IN 1..6 LOOP
      UPDATE la_crimes_temp
      SET crime_location = TRIM(REPLACE(crime_location, '  ', ' ')),
         cross_street = TRIM(REPLACE(cross_street, '  ', ' '));
      PERFORM crime_location, cross_street
      FROM la_crimes_temp
      WHERE cross_street IS NOT NULL;
   END LOOP;
END $$;
```

- Above is a PL/SQL FOR LOOP within a DO statement to use the previous update statement.
- 6 loops are what has shown to remove all excess spaces. If this changes the number of loops will be increased.

```
--- Creating loads of temp tables into actual tables
INSERT INTO crimes
SELECT * FROM crimes_temp
ON CONFLICT (crime_code) DO NOTHING;
```

```sql
-- Load data into the `crime_weapons` table
INSERT INTO crime_weapons
SELECT * FROM crime_weapons_temp
ON CONFLICT (weapon_used_code) DO NOTHING;

-- Load data into the `case_details` table
INSERT INTO case_details
SELECT * FROM case_details_temp
ON CONFLICT (case_status) DO NOTHING;

-- Load data into the `premises` table
INSERT INTO premises
SELECT * FROM premises_temp
ON CONFLICT (premises_code) DO NOTHING;

-- Load data into the `crime_geos` table
INSERT INTO crime_geos
SELECT * FROM crime_geos_temp
ON CONFLICT (geo_area_num) DO NOTHING;

-- Load data into the `crime_victim_descents` table
INSERT INTO crime_victim_descents
SELECT * FROM crime_victim_descents_temp
ON CONFLICT (victim_descent) DO NOTHING;

-- Load data into the `crime_codes` table
INSERT INTO crime_codes
SELECT * FROM crime_codes_temp
ON CONFLICT (division_records_num) DO NOTHING;

-- Load data into the `date_table` table
INSERT INTO date_table
SELECT dt.*
FROM date_table_temp dt
LEFT JOIN date_table d ON dt.date = d.date
WHERE d.date IS NULL;

-- Load data into the `la_crimes` table
INSERT INTO la_crimes
```

```
SELECT * FROM la_crimes_temp
ON CONFLICT (division_records_num) DO NOTHING;
```

- The provided code uses a series of INSERT statements to transfer data from temporary tables into the original tables.
- The ON CONFLICT (primary key) DO NOTHING clause ensures that rows with duplicate primary keys already existing in the base table are ignored, preventing conflicts while maintaining the integrity of the data in the temp tables.