SQL Code Summary for Mobile Phone Market Analysis

This summary provides a straightforward overview of the SQL code used to analyze a dataset of mobile phones. The steps cover data preparation, cleaning, and analysis to derive insights for marketing, pricing, and product development.

1. **Creating the Mobile Data Table**:
   A table named `mobile_data` is created to store mobile phone details such as `name`, `battery`, `camera`, `display`, `processor`, `memory`, `warranty`, `price`, `rating`, and `reviews`.
2. **Uploading Data**:
   The data from a CSV file (`mobile_dataset.csv`) is loaded into the `mobile_data` table. The file path, delimiter (`,`), and header information are specified in the `COPY` command to ensure proper data import.
3. **Exploring Data**:
   The `SELECT` statement retrieves all records from the `mobile_data` table, sorted by `price` in ascending order. This helps in reviewing the dataset and understanding the range of prices.
4. **Counting Total Records**:
   The `COUNT(*)` function is used to find the total number of records in the `mobile_data` table, giving an idea of the dataset size.
5. **Checking for Missing Values**:
   The SQL code calculates the number of missing values in each column by using `COUNT` and `CASE` statements. This identifies columns that have incomplete or missing data marked as 'Null'.
6. **Cleaning the Data**:
   A `DELETE` command removes all rows where any column (like `name`, `battery`, `camera`, etc.) contains 'Null'. This step ensures the data is clean and consistent for accurate analysis.
7. **Analyzing Popular Configurations**:
   - **Battery Types**: The data is grouped by battery type to count and identify the most popular battery configurations.
   - **Camera Setups**: Similar grouping and counting is done for camera setups to find the most common options.
   - **Display Types**: The dataset is grouped by display type to understand the distribution across different phones.
   - **Processors**: Grouping by processor type helps in identifying which processors are most commonly used.
8. **Price Segmentation (Clustering)**:
   Although the SQL code does not explicitly define price clustering, it suggests dividing phones into different price segments (e.g., low, mid, high). This analysis helps in

understanding how price categories relate to other features like battery, camera, and processor.

Key Takeaways:

- The SQL code provides a step-by-step approach to cleaning, exploring, and analyzing a mobile phone dataset.
- Grouping by key attributes (battery, camera, display, processor) helps in understanding popular configurations and customer preferences.
- Removing rows with missing values ensures a clean dataset, improving the reliability of further analysis.
- Segmenting by price range can reveal valuable insights for developing marketing strategies, optimizing pricing, and tailoring product offerings to customer needs.

This analysis forms a solid foundation for understanding the mobile phone market dynamics and identifying factors that influence buying decisions.

```
CREATE TABLE mobile_data (
   name varchar(100),
        battery varchar(100),
        camera varchar(100),
        display varchar(100),
        processor        varchar(100),
        memory varchar(100),
        warranty varchar(100),
        price varchar(100),
        rating varchar(100),
        reviews Numeric
        );
--display table
select *
from mobile_data
order by price asc;

--uploading csv file
COPY mobile_data(name, battery, camera, display, processor, memory, warranty, price, rating, reviews)
FROM 'C:\Users\shikh\Desktop\JA_Assignment\mobile_dataset.csv'
DELIMITER ','
CSV HEADER;
--count no of data avaiable
select count(*)
from mobile_data;
--no of null values
SELECT
   COUNT(*) AS total_records,
```

```
    COUNT(CASE WHEN name = 'Null' THEN 1 END) AS name_null_count,
    COUNT(CASE WHEN battery = 'Null' THEN 1 END) AS battery_null_count,
    COUNT(CASE WHEN camera = 'Null' THEN 1 END) AS camera_null_count,
    COUNT(CASE WHEN display = 'Null' THEN 1 END) AS display_null_count,
    COUNT(CASE WHEN processor = 'Null' THEN 1 END) AS processor_null_count,
    COUNT(CASE WHEN memory = 'Null' THEN 1 END) AS memory_null_count,
    COUNT(CASE WHEN warranty = 'Null' THEN 1 END) AS warranty_null_count,
    COUNT(CASE WHEN price = 'Null' THEN 1 END) AS price_null_count,
    COUNT(CASE WHEN rating = 'Null' THEN 1 END) AS rating_null_count,
    COUNT(CASE WHEN reviews = 'Null' THEN 1 END) AS reviews_null_count
FROM mobile_data;
--delete any null rows
DELETE FROM mobile_data
WHERE name = 'Null'
   OR battery = 'Null'
   OR camera = 'Null'
   OR display = 'Null'
   OR processor = 'Null'
   OR memory = 'Null'
   OR warranty = 'Null'
   OR price = 'Null'
   OR rating = 'Null'
   OR reviews = 'Null';
--categorising battery, processor, memory, display
SELECT battery, COUNT(*) AS count
FROM mobile_data
GROUP BY battery
ORDER BY count DESC;

SELECT camera, COUNT(*) AS count
FROM mobile_data
GROUP BY camera
ORDER BY count DESC;

SELECT display, COUNT(*) AS count
FROM mobile_data
GROUP BY display
ORDER BY count DESC;

SELECT processor, COUNT(*) AS count
FROM mobile_data
GROUP BY processor
ORDER BY count DESC;

--price clusterring
```