Exercise: Create a Sales Data Warehouse Table in BigQuery

Objectives

- Understand and apply the concept of querying a Data Warehouse (DWH) table in SQL.
- Complete a skeleton SQL statement to create and query a DWH table in BigQuery.
- Create charts in Looker Studio to visualize the results of a SQL query.

Instructions

Part 1: Creating a Sales Data Warehouse Table

You will create a Data Warehouse table named `dwh_sales_data` in the `adventureworks` dataset. The table should be created using CTEs to organize the data from various source tables.

Before beginning, update the script for the `dwh_product_with_reviews` table and convert string fields to numbers where appropriate.

```
 CREATE OR REPLACE TABLE `adventureworks.dwh_product_with_reviews` AS
 ...
    CAST(p.standardcost AS NUMERIC) AS standardcost,
    CAST(p.listprice AS NUMERIC) AS listprice,
...
```

Then, create a new DWH table

Skeleton:

```
CREATE OR REPLACE TABLE `adventureworks.dwh_sales_data` AS
WITH
  salesorder AS (
    -- Complete the CTE definition here; join salesorderheader,
     ↪  salesorderdetail and dwh_product_with_reviews tables
  )
SELECT
  -- Specify the columns to be included in the final table
FROM
  salesorder;
```

Your Task: Using the provided skeleton, complete the CTE and SELECT statement to create the `dwh_sales_data` table with the specified columns from the `salesorderdetail`, `salesorderheader` and `dwh_product_with_reviews` tables. **Don't forget to convert STRINGS to NUMERIC where appropriate.**

Part 2: Revenue Analysis Query

Your task is to write a SQL query that compares various revenue metrics for each year. You will calculate the actual revenue, potential revenue at list price, total discounts given, and revenue loss from the list price.

Example:

```
SUM(unitprice * orderqty * unitpricediscount) AS total_discount
```

- unitpriice –> actual price
- listprice –> price for potential revenue
- unitpricediscount –> e.g. 0.4 –> 40% discount
- orderqty –> number of items sold at that price
- loss –> listprice - unitprice

Skeleton:

```
SELECT
  EXTRACT(YEAR FROM orderdate) AS year,
  -- Calculate actual revenue
  -- Calculate potential revenue at list price
  -- Calculate total discount given
  -- Calculate revenue loss from list price
FROM
  `adventureworks.dwh_sales_data`
GROUP BY
  year
ORDER BY
  year;
```

Your Task: Complete the query to calculate the required revenue metrics. **Once the query is executed, use Looker Studio to create a bar chart visualizing these metrics by year.**

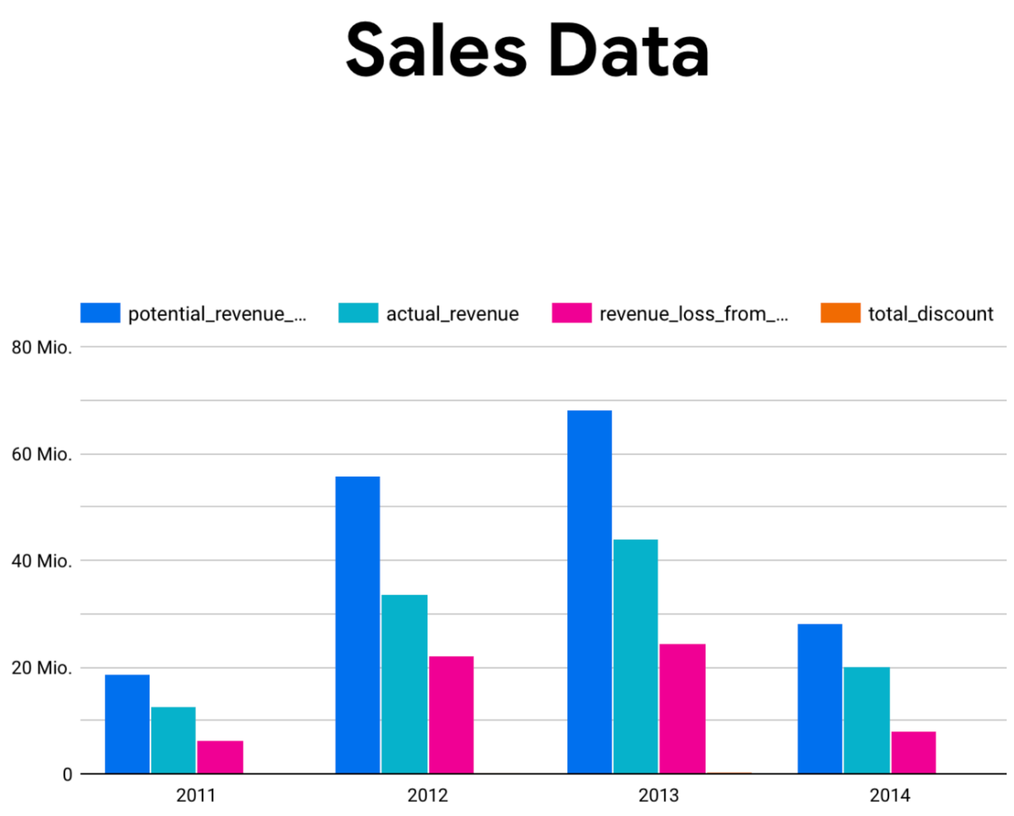

Figure 1: Sales Data

Part 3: Understanding Binning in SQL

Binning, or bucketing, is a process of dividing continuous data into discrete ranges, which helps in data categorization and summarization.

Syntax:

```
SELECT
  CASE
    WHEN condition THEN 'Label'
    -- Additional conditions
    ELSE 'OtherLabel'
  END AS bin_column,
  COUNT(*) AS metric
FROM table
GROUP BY bin_column;
```

Example: To create bins for age groups, you might categorize ages into 'Youth', 'Adult', and 'Senior':

```sql
SELECT
  CASE
    WHEN age < 18 THEN 'Youth'
    WHEN age BETWEEN 18 AND 64 THEN 'Adult'
    ELSE 'Senior'
  END AS AgeGroup,
  COUNT(*) AS Count
FROM People
GROUP BY AgeGroup;
```

Your task is to categorize products into different price ranges and then visualize the distribution of products across these ranges using Looker Studio.

Skeleton:

```sql
SELECT
  CASE
    -- Define price range conditions
  END AS price_range,
  COUNT(*) AS product_count
FROM
  `adventureworks.dwh_product_with_reviews`
GROUP BY
  price_range
ORDER BY
  price_range;
```

Your Task: Complete the query to create bins for different product price ranges. **Then, use Looker Studio to create a visualization that shows the distribution of products in each price range.**

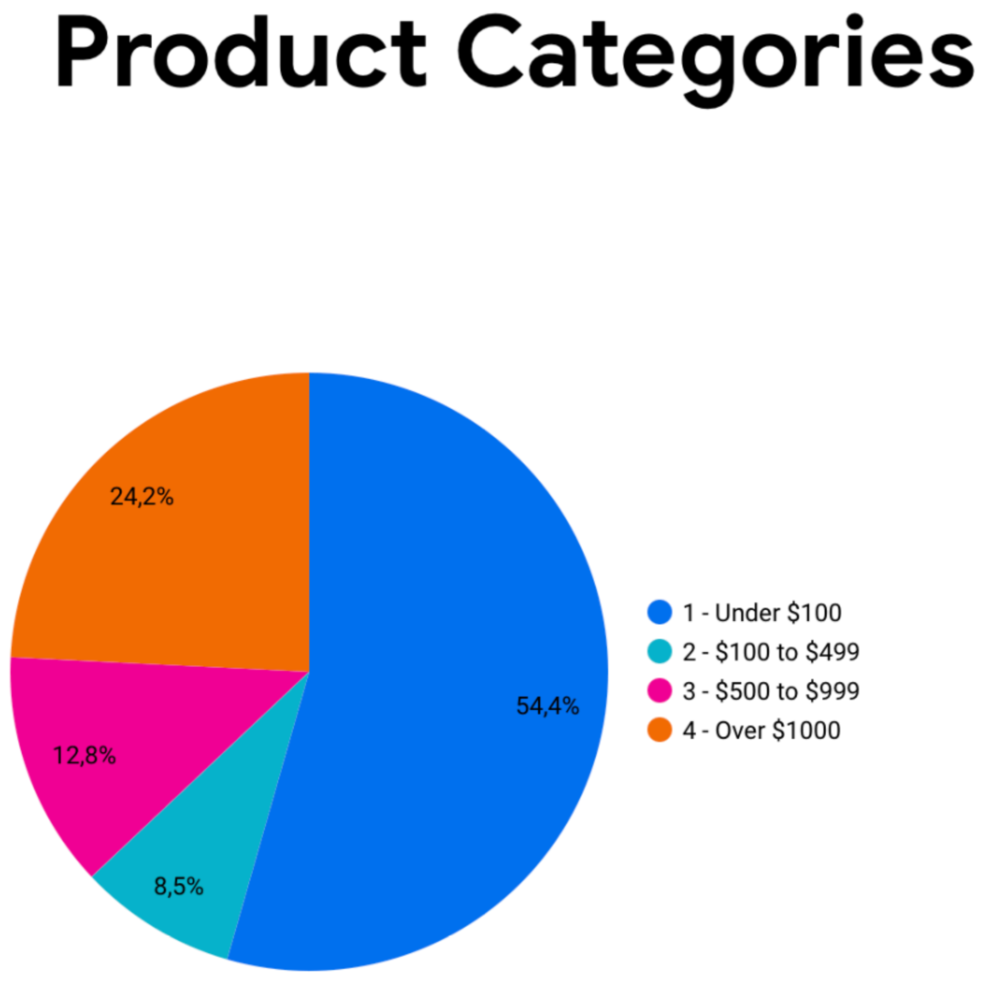

Figure 2: Product Categories