---

## 1️⃣ Centralized Age Groups

```
SELECT AgeGroupID, AgeGroupName
FROM `familysmsdev.retail_sales_snowflake_schema.age_group_lookup`
ORDER BY AgeGroupID;
```

**Advantage:** Update an age group label once and all related customer records automatically inherit the change.

---

## 2️⃣ Customers with County

```
SELECT c.CustomerName,
       co.CountyName
FROM `familysmsdev.retail_sales_snowflake_schema.customer_dim` AS c
JOIN `familysmsdev.retail_sales_snowflake_schema.town_lookup`  AS t
  ON c.TownID = t.TownID
JOIN `familysmsdev.retail_sales_snowflake_schema.county_lookup` AS co
  ON t.CountyID = co.CountyID
LIMIT 20;
```

**Advantage:** County names live in one place—no spelling inconsistencies across the data model.

---

## 3️⃣ Revenue by County

```
SELECT co.CountyName,
       SUM(sf.Revenue) AS TotalRevenue
FROM `familysmsdev.retail_sales_snowflake_schema.sales_fact` AS sf
JOIN `familysmsdev.retail_sales_snowflake_schema.store_dim` AS s
  ON sf.StoreID = s.StoreID
JOIN `familysmsdev.retail_sales_snowflake_schema.town_lookup` AS t
  ON s.TownID = t.TownID
JOIN `familysmsdev.retail_sales_snowflake_schema.county_lookup` AS co
  ON t.CountyID = co.CountyID
```

```
GROUP BY co.CountyName
ORDER BY TotalRevenue DESC;
```

**Advantage:** Multi-level joins (store → town → county) are straightforward because each level is normalized.

---

## 4️⃣ Revenue by Product Category

```
SELECT cat.CategoryName,
       SUM(sf.Revenue) AS CategoryRevenue
FROM `familysmsdev.retail_sales_snowflake_schema.sales_fact` AS sf
JOIN `familysmsdev.retail_sales_snowflake_schema.product_dim` AS p
  ON sf.ProductID = p.ProductID
JOIN `familysmsdev.retail_sales_snowflake_schema.category_lookup` AS cat
  ON p.CategoryID = cat.CategoryID
GROUP BY cat.CategoryName
ORDER BY CategoryRevenue DESC;
```

**Advantage:** Category definitions are maintained centrally for easy reclassification.

---

## 5️⃣ Revenue by Brand

```
SELECT b.BrandName,
       SUM(sf.Revenue) AS TotalRevenue
FROM `familysmsdev.retail_sales_snowflake_schema.sales_fact` AS sf
JOIN `familysmsdev.retail_sales_snowflake_schema.product_dim` AS p
  ON sf.ProductID = p.ProductID
JOIN `familysmsdev.retail_sales_snowflake_schema.brand_lookup` AS b
  ON p.BrandID = b.BrandID
GROUP BY b.BrandName
ORDER BY TotalRevenue DESC;
```

**Advantage:** Brand data stored once—brand renames or mergers require only one update.

---

## 6️⃣ Quarterly Revenue Trend

```
SELECT q.QuarterName,
       SUM(sf.Revenue) AS TotalRevenue
FROM `familysmsdev.retail_sales_snowflake_schema.sales_fact` AS sf
JOIN `familysmsdev.retail_sales_snowflake_schema.time_dim` AS td
  ON sf.TimeID = td.TimeID
JOIN `familysmsdev.retail_sales_snowflake_schema.quarter_lookup` AS q
  ON td.QuarterID = q.QuarterID
GROUP BY q.QuarterName
ORDER BY q.QuarterName;
```

**Advantage:** Fiscal quarter changes only require updating the `quarter_lookup` table.

---

## 7️⃣ Revenue by Gender

```
SELECT g.GenderName,
       SUM(sf.Revenue) AS TotalRevenue
FROM `familysmsdev.retail_sales_snowflake_schema.sales_fact` AS sf
JOIN `familysmsdev.retail_sales_snowflake_schema.customer_dim` AS c
  ON sf.CustomerID = c.CustomerID
JOIN `familysmsdev.retail_sales_snowflake_schema.gender_lookup` AS g
  ON c.GenderID = g.GenderID
GROUP BY g.GenderName;
```

**Advantage:** Gender categories can evolve without altering the fact table.

---

## 8️⃣ Units Sold by Age Group

```
SELECT ag.AgeGroupName,
       SUM(sf.UnitsSold) AS TotalUnits
FROM `familysmsdev.retail_sales_snowflake_schema.sales_fact` AS sf
JOIN `familysmsdev.retail_sales_snowflake_schema.customer_dim` AS c
  ON sf.CustomerID = c.CustomerID
JOIN `familysmsdev.retail_sales_snowflake_schema.age_group_lookup` AS ag
  ON c.AgeGroupID = ag.AgeGroupID
```

```
GROUP BY ag.AgeGroupName
ORDER BY TotalUnits DESC;
```

**Advantage:** Demographic analysis is easy because age groups are centrally defined.

---

## 9️⃣ Manager Performance

```
SELECT m.ManagerName,
       SUM(sf.Revenue) AS ManagerRevenue
FROM `familysmsdev.retail_sales_snowflake_schema.sales_fact` AS sf
JOIN `familysmsdev.retail_sales_snowflake_schema.store_dim` AS s
  ON sf.StoreID = s.StoreID
JOIN `familysmsdev.retail_sales_snowflake_schema.manager_lookup` AS m
  ON s.ManagerID = m.ManagerID
GROUP BY m.ManagerName
ORDER BY ManagerRevenue DESC;
```

**Advantage:** Manager information is stored once—reassignments or name corrections require a single update.

---

## 🔟 Data Integrity Check for Orphan Towns

```
SELECT c.CustomerID,
       c.CustomerName,
       c.TownID
FROM `familysmsdev.retail_sales_snowflake_schema.customer_dim` AS c
LEFT JOIN `familysmsdev.retail_sales_snowflake_schema.town_lookup` AS t
  ON c.TownID = t.TownID
WHERE t.TownID IS NULL;
```

**Advantage:** Normalized keys make it simple to find missing or invalid references.

---

### Key Snowflake-Schema Advantages Demonstrated

- **Centralized attributes** – easy maintenance and consistent values.
- **Reduced redundancy** – fact table stays small; no repeated text.
- **Flexible analysis** – straightforward drill-down across multiple normalized layers.
- **Data integrity** – foreign keys allow quick integrity checks and orphan detection.