**Subject:** Review of Satellite Dimension Approach in *ARC Model PoC 4*

Hi [Colleague’s Name],

I’ve been experimenting with a **satellite dimensioning approach** in our semantic model (*ARC Model PoC 4*) to assess whether it can improve DirectQuery performance.

As part of this prototype, I split the existing dimension **pbi\_synpoc\_dim\_risk\_factor1** into two related tables:

**pbi\_synpoc\_dim\_risk\_factor1\_ext1** – directly related to the fact table **pbi\_synpoc\_fact\_risk\_results**

**pbi\_synpoc\_dim\_risk\_factor1\_ext2** – related to **pbi\_synpoc\_dim\_risk\_factor1\_ext1**, not directly to the fact

The goal is to apply a *satellite dimension design*—a controlled form of snowflaking—to isolate heavy or rarely used attributes into a secondary table while keeping the high-value attributes close to the fact.

### **Rationale for the Split**

**Cardinality Optimization:**  
 Columns with very high distinct counts can slow down query execution, especially in slicers or group-by operations. By separating low-cardinality (frequently used) attributes from high-cardinality (detailed or descriptive) ones, we aim to reduce the number of DISTINCT scans and shrink the size of the dimension directly joining the fact table.

**Usage Pattern Separation:**

**Ext1** contains attributes that are often used in **slicers, filters, and relationships**, where lower latency is critical.

**Ext2** holds **granular or display-only columns**—for example, long text, descriptive names, or rarely used attributes—that can be loaded on demand without impacting everyday queries.

**Model Simplicity and Performance:**  
 By having only one active relationship (Fact → Ext1), we keep the model simple, reduce join ambiguity, and allow Power BI and Databricks to perform better column and file pruning. The secondary relationship (Ext1 → Ext2) is used only when those extra attributes are requested, so the DirectQuery workload should be lighter in typical use cases.

Although early tests show performance close to the original single-table setup, this approach might yield more benefit once physical optimizations are applied (for example, clustering or Z-ORDERing by the key, better caching, and focused slicer usage).

Could you please review this implementation and evaluate the **feasibility and expected performance gains** of this satellite dimension structure? Your feedback will help decide whether we should extend this pattern to other large dimensions.

Thanks,  
 **Julio**