Data Schema and Models

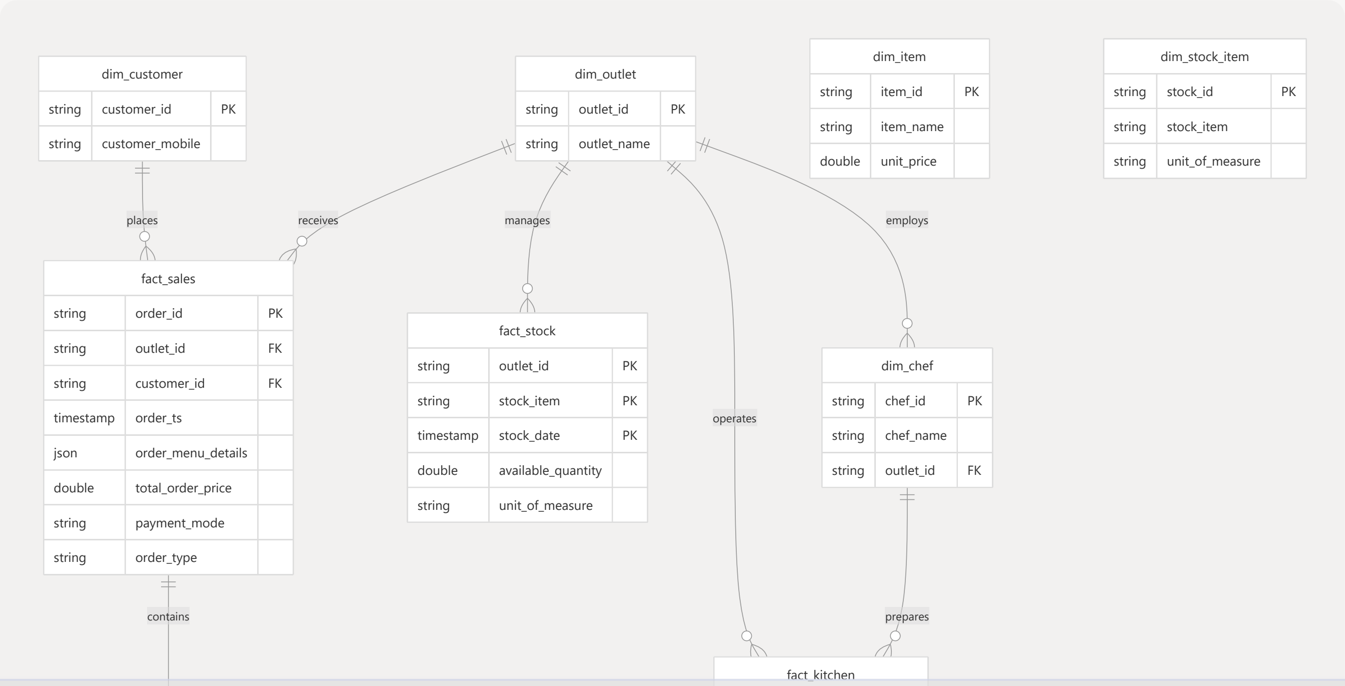
Relevant source files

This document provides a comprehensive reference for all dimension and fact tables in the Restaurant data processing system. It covers base schemas used for data generation and ingestion, validation configurations, and enriched schemas produced by the harmonization pipeline.

For information about the data generation process, see Data Generation. For the source fact ingestion pipeline that validates and loads these tables, see Source Facts Pipeline. For the enrichment logic that produces derived schemas, see Fact Enrichment.

Overview

The Restaurant system implements a **star schema** data model with five dimension tables and three fact tables. The model captures restaurant operations including orders, kitchen activities, and inventory management.



dim_customer		
string	customer_id	PK
string	customer_mobile	

dim_outlet		
string	outlet_id	PK
string	outlet_name	

dim_item		
string	item_id	PK
string	item_name	
double	unit_price	

dim_stock_item		
string	stock_id	PK
string	stock_item	
string	unit_of_measure	

fact_sales		
string	order_id	PK
string	outlet_id	FK
string	customer_id	FK
timestamp	order_ts	
json	order_menu_details	
double	total_order_price	
string	payment_mode	
string	order_type	

fact_stock		
string	outlet_id	PK
string	stock_item	PK
timestamp	stock_date	PK
double	available_quantity	
string	unit_of_measure	

dim_chef		
string	chef_id	PK
string	chef_name	
string	outlet_id	FK

fact_kitchen		
string	order_id	PK
string	outlet_id	FK
string	chef_id	FK
timestamp	cooking_start	
timestamp	cooking_end	

fact_sales_items

Ask Devin about rahulbhassy/Restaurant

Fast

Sources: `Resturant/DataGenerator/data_generator.py 23-179` `Resturant/DataGenerator/schema.py 1-103`

Dimension Tables

dim_customer

Stores customer master data with mobile contact information.

Column	Type	Nullable	Description
`customer_id`	StringType	False	Primary key (format: CUST0001-CUST9999)
`customer_mobile`	StringType	True	Phone number (+91 prefix)

Schema Definition: `Resturant/DataGenerator/schema.py 9-12`

Validation Rules:

- **Mandatory columns:** `customer_id`
- **Duplicate key:** `customer_id`
- **No type casts, allowed values, or anomaly rules**

Sources: `Resturant/DataGenerator/data_generator.py 55-63` `Resturant/SourceFact/config.py 3-9`

dim_outlet

Defines restaurant outlet locations.

Column	Type	Nullable	Description
`outlet_id`	StringType	False	Primary key (format: OUT001-OUT005)
`outlet_name`	StringType	True	Outlet name (e.g., "KFC Koramangala")

Schema Definition: `Resturant/DataGenerator/schema.py 20-23`

Validation Rules:

- **Mandatory columns:** `outlet_id`
- **Duplicate key:** `outlet_id`

Sample Data:

```
OUT001 → KFC Koramangala
OUT002 → KFC Indiranagar
OUT003 → KFC Whitefield
OUT004 → KFC MG Road
OUT005 → KFC Electronic City
```

Sources: `Resturant/DataGenerator/data_generator.py` 16-22 `Resturant/SourceFact/config.py` 19-25

dim_chef

Contains chef master data with outlet assignments.

Column	Type	Nullable	Description
`chef_id`	StringType	False	Primary key (format: CHEF_OUT001_1)
`chef_name`	StringType	True	Chef display name
`outlet_id`	StringType	True	Foreign key to dim_outlet

Schema Definition: `Resturant/DataGenerator/schema.py` 25-29

Validation Rules:

- **Mandatory columns:** `chef_id`
- **Duplicate key:** `chef_id`

Generation Logic: Each outlet receives 3-6 randomly assigned chefs during data generation.

Sources: `Resturant/DataGenerator/data_generator.py` 66-76 `Resturant/SourceFact/config.py` 27-33

dim_item

Menu item catalog with pricing.

Column	Type	Nullable	Description
`item_id`	StringType	False	Primary key (format: ITEM001-ITEM010)
`item_name`	StringType	True	Menu item name
`unit_price`	DoubleType	True	Price per unit

Schema Definition: `Resturant/DataGenerator/schema.py` 14-18

Validation Rules:

- **Mandatory columns:** `item_id`
- **Type casting:** `unit_price` → double
- **Duplicate key:** `item_id`
- **Anomaly detection:** `unit_price` > 10000

Sample Items:

```
ITEM001 → Zinger Burger (₹180)
ITEM002 → Hot & Crispy Chicken (₹220)
ITEM003 → Chicken Popcorn (₹150)
ITEM008 → Pepsi (₹60)
```

Sources: `Resturant/DataGenerator/data_generator.py` 27-38

`Resturant/SourceFact/config.py` 11-17

dim_stock_item

Inventory item catalog with units of measure.

Column	Type	Nullable	Description
`stock_id`	StringType	False	Primary key (format: ST001-ST009)
`stock_item`	StringType	True	Inventory item name
`unit_of_measure`	StringType	True	Unit (kg, g, litre, piece)

Schema Definition: `Resturant/DataGenerator/schema.py` 31-35

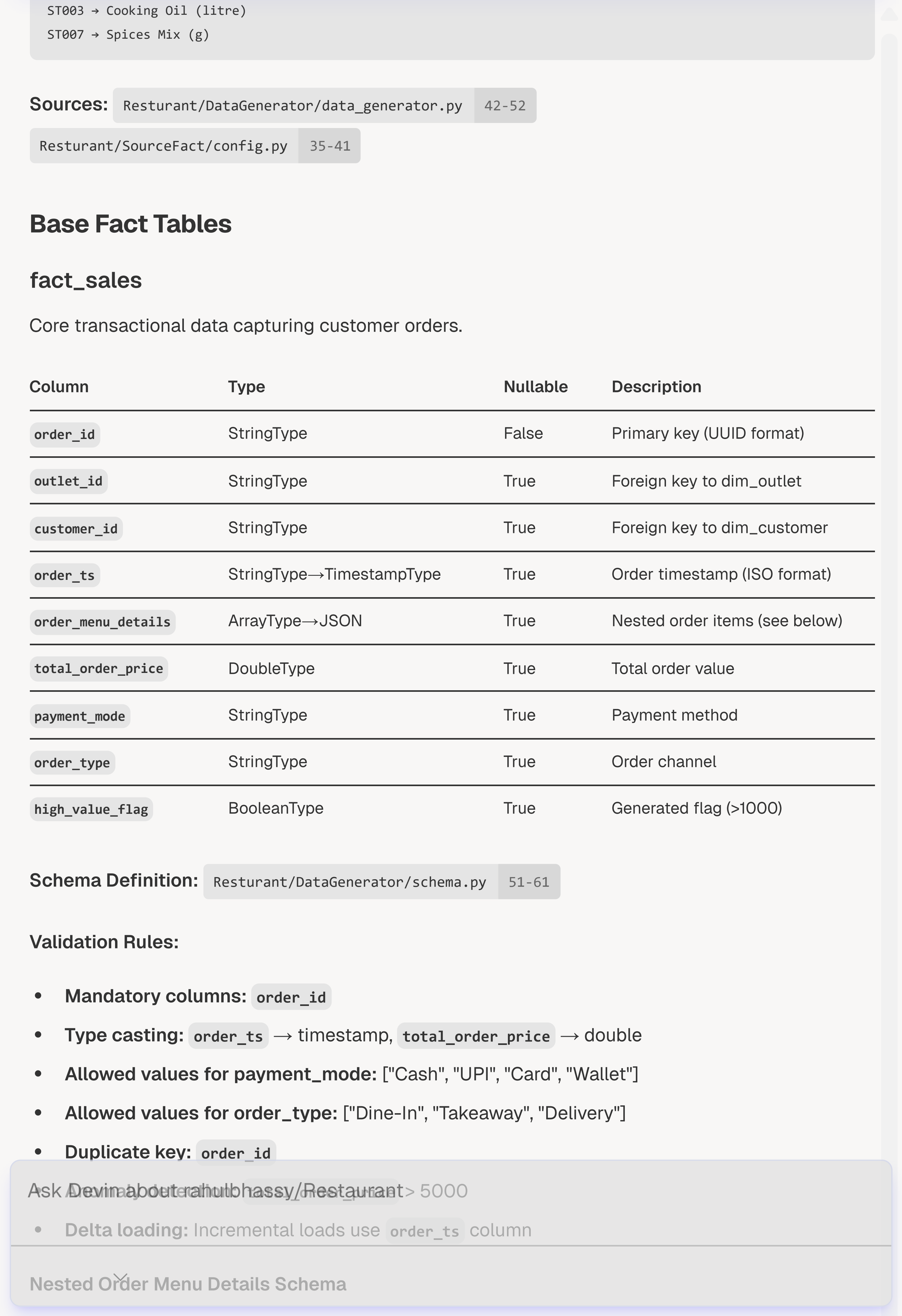
Validation Rules:

- **Mandatory columns:** `stock_id`
- **Duplicate key:** `stock_id`
- **Allowed values for unit_of_measure:** ["kg", "g", "litre", "piece"]

Sample Items:

Ask Devin about rahulbhassy/Restaurant

```
ST001 → Chicken Breast (kg)
ST002 → Burger Buns (piece)
```



```
ST003 → Cooking Oil (litre)
ST007 → Spices Mix (g)
```

Sources: `Resturant/DataGenerator/data_generator.py` 42-52 `Resturant/SourceFact/config.py` 35-41

Base Fact Tables

fact_sales

Core transactional data capturing customer orders.

Column	Type	Nullable	Description
`order_id`	StringType	False	Primary key (UUID format)
`outlet_id`	StringType	True	Foreign key to dim_outlet
`customer_id`	StringType	True	Foreign key to dim_customer
`order_ts`	StringType→TimestampType	True	Order timestamp (ISO format)
`order_menu_details`	ArrayType→JSON	True	Nested order items (see below)
`total_order_price`	DoubleType	True	Total order value
`payment_mode`	StringType	True	Payment method
`order_type`	StringType	True	Order channel
`high_value_flag`	BooleanType	True	Generated flag (>1000)

Schema Definition: `Resturant/DataGenerator/schema.py` 51-61

Validation Rules:

- **Mandatory columns:** `order_id`
- **Type casting:** `order_ts` → timestamp, `total_order_price` → double
- **Allowed values for payment_mode:** ["Cash", "UPI", "Card", "Wallet"]
- **Allowed values for order_type:** ["Dine-In", "Takeaway", "Delivery"]
- **Duplicate key:** `order_id`
- **Anomaly detection:** `total_order_price` > 5000
- **Delta loading:** Incremental loads use `order_ts` column

Ask Devin about rahulbhassy/Restaurant

Nested Order Menu Details Schema

The `order_menu_details` field contains a JSON-serialized array with this structure:

```
[
  {
    "item_id": "ITEM001",
    "item_name": "Zinger Burger",
    "quantity": 2,
    "toppings": ["Extra Cheese", "Spicy Sauce"],
    "unit_price": 180.0,
    "line_total": 360.0
  }
]
```

Nested Schema Definition: `Resturant/DataGenerator/schema.py` 42-49

Sources: `Resturant/DataGenerator/data_generator.py` 138-148 `Resturant/SourceFact/config.py` 43-55 `Resturant/DataGenerator/data_ingester.py` 31-32

fact_kitchen

Kitchen operations data tracking cooking times and chef assignments.

Column	Type	Nullable	Description
`order_id`	StringType	False	Primary key, foreign key to fact_sales
`outlet_id`	StringType	True	Foreign key to dim_outlet
`chef_id`	StringType	True	Foreign key to dim_chef
`cooking_start`	StringType→TimestampType	True	Cooking start timestamp
`cooking_end`	StringType→TimestampType	True	Cooking end timestamp

Schema Definition: `Resturant/DataGenerator/schema.py` 63-69

Validation Rules:

- **Mandatory columns:** `order_id`
- **Type casting:** `cooking_start` → timestamp, `cooking_end` → timestamp
- **Duplicate key:** `order_id`
- **Delta loading:** Incremental loads use `cooking_end` column
- **Merge key:** `order_id` for UPSERT operations

Ask Devin about rahulbhassy/Restaurant

Sources: `Resturant/DataGenerator/data_generator.py` 150-157

`Resturant/SourceFact/config.py` 57-66

fact_stock

Inventory snapshot data tracking stock levels by outlet and date.

Column	Type	Nullable	Description
`outlet_id`	StringType	True	Foreign key to dim_outlet
`stock_item`	StringType	True	Item name (links to dim_stock_item)
`available_quantity`	DoubleType	True	Current quantity on hand
`unit_of_measure`	StringType	True	Unit (kg, g, litre, piece)
`stock_date`	StringType→TimestampType	True	Snapshot date

Schema Definition: `Resturant/DataGenerator/schema.py` 71-77

Validation Rules:

- **Mandatory columns:** `outlet_id`, `stock_item`
- **Type casting:** `stock_date` → timestamp, `available_quantity` → double
- **Allowed values for unit_of_measure:** ["kg", "g", "litre", "piece"]
- **Duplicate keys:** [`outlet_id`, `stock_item`, `stock_date`]
- **Anomaly detection:** `available_quantity` < -1 (negative inventory)
- **No delta loading:** Always full refresh

Sources: `Resturant/DataGenerator/data_generator.py` 97-104

`Resturant/SourceFact/config.py` 68-77

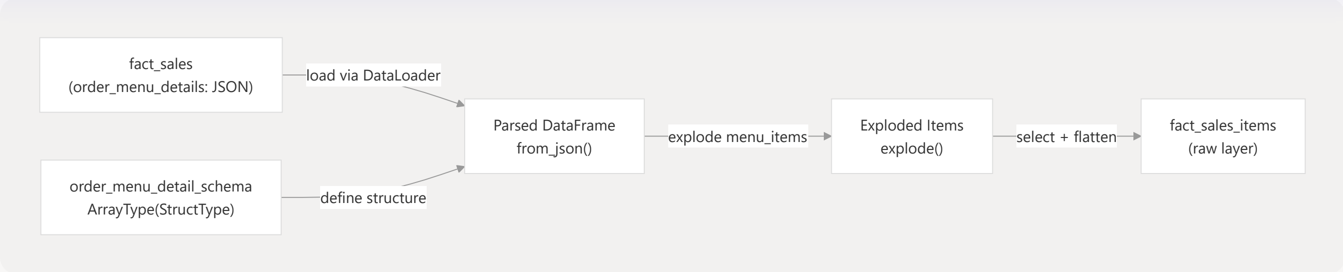
Derived Fact Tables

The enrichment pipeline produces three derived fact tables with additional analytical attributes. These tables extend base facts with time dimensions, aggregations, and business metrics.

fact_sales_items

Ask Devin about rahulbhassy/Restaurant

Flattened item-level view extracted from `fact_sales.order_menu_details` JSON.



Column	Type	Description
`order_id`	StringType	Foreign key to fact_sales
`order_ts`	TimestampType	Order timestamp
`outlet_id`	StringType	Foreign key to dim_outlet
`customer_id`	StringType	Foreign key to dim_customer
`item_id`	StringType	Menu item identifier
`item_name`	StringType	Menu item name
`quantity`	IntegerType	Quantity ordered
`unit_price`	DoubleType	Price per unit
`line_total`	DoubleType	Line-level total
`toppings`	ArrayType(StringType)	Customizations
`ingested_at`	TimestampType	Processing timestamp

Creation Logic: `Resturant/EnrichFact/Harmonisation.py` `84-106`

Process Flow:

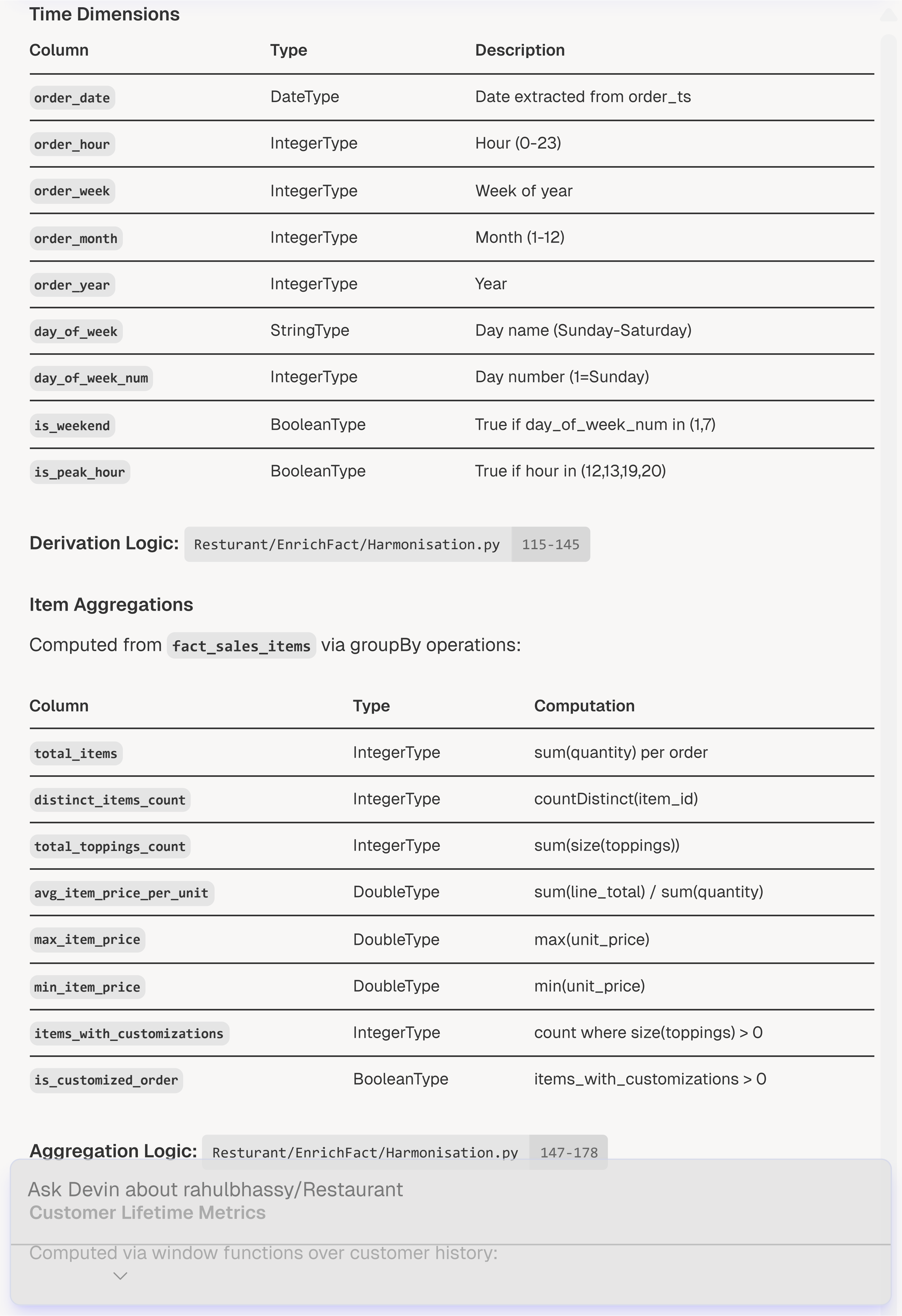
1. Load `fact_sales` from raw layer `Harmonisation.py` `67-73`
2. Parse JSON using `from_json()` with schema `Harmonisation.py` `56-65`
3. Explode array to create one row per item `Harmonisation.py` `91`
4. Flatten nested structure and select columns `Harmonisation.py` `92-103`
5. Write to raw layer as Delta table `Harmonisation.py` `75-82`

Sources: `Resturant/EnrichFact/Harmonisation.py` `34-107`

Ask Devin about rahulbhassy/Restaurant

fact_sales_enriched

Comprehensive enriched sales fact with 30+ derived analytical attributes.



Time Dimensions

Column	Type	Description
`order_date`	DateType	Date extracted from order_ts
`order_hour`	IntegerType	Hour (0-23)
`order_week`	IntegerType	Week of year
`order_month`	IntegerType	Month (1-12)
`order_year`	IntegerType	Year
`day_of_week`	StringType	Day name (Sunday-Saturday)
`day_of_week_num`	IntegerType	Day number (1=Sunday)
`is_weekend`	BooleanType	True if day_of_week_num in (1,7)
`is_peak_hour`	BooleanType	True if hour in (12,13,19,20)

Derivation Logic: `Resturant/EnrichFact/Harmonisation.py` 115-145

Item Aggregations

Computed from `fact_sales_items` via groupBy operations:

Column	Type	Computation
`total_items`	IntegerType	sum(quantity) per order
`distinct_items_count`	IntegerType	countDistinct(item_id)
`total_toppings_count`	IntegerType	sum(size(toppings))
`avg_item_price_per_unit`	DoubleType	sum(line_total) / sum(quantity)
`max_item_price`	DoubleType	max(unit_price)
`min_item_price`	DoubleType	min(unit_price)
`items_with_customizations`	IntegerType	count where size(toppings) > 0
`is_customized_order`	BooleanType	items_with_customizations > 0

Aggregation Logic: `Resturant/EnrichFact/Harmonisation.py` 147-178

Ask Devin about rahulbhassy/Restaurant

Customer Lifetime Metrics

Computed via window functions over customer history:

Column	Type	Description
`customer_mobile`	StringType	Joined from dim_customer
`total_orders_by_customer`	LongType	Lifetime order count
`customer_lifetime_value`	DoubleType	sum(total_order_price)
`is_repeat_customer`	BooleanType	total_orders_by_customer > 1
`days_since_last_order`	LongType	Date diff from max(order_ts)

Computation Logic: `Resturant/EnrichFact/Harmonisation.py` 180-201 `Harmonisation.py` 273-282

Kitchen Performance Metrics

Joined from `fact_kitchen` data:

Column	Type	Description
`cook_duration_seconds`	LongType	cooking_end - cooking_start
`was_delayed_order`	BooleanType	cook_duration_seconds > 600 (SLA)

Computation Logic: `Resturant/EnrichFact/Harmonisation.py` 203-221

SLA Configuration: 600 seconds (10 minutes) defined at `Harmonisation.py` 111

Derived Business Flags

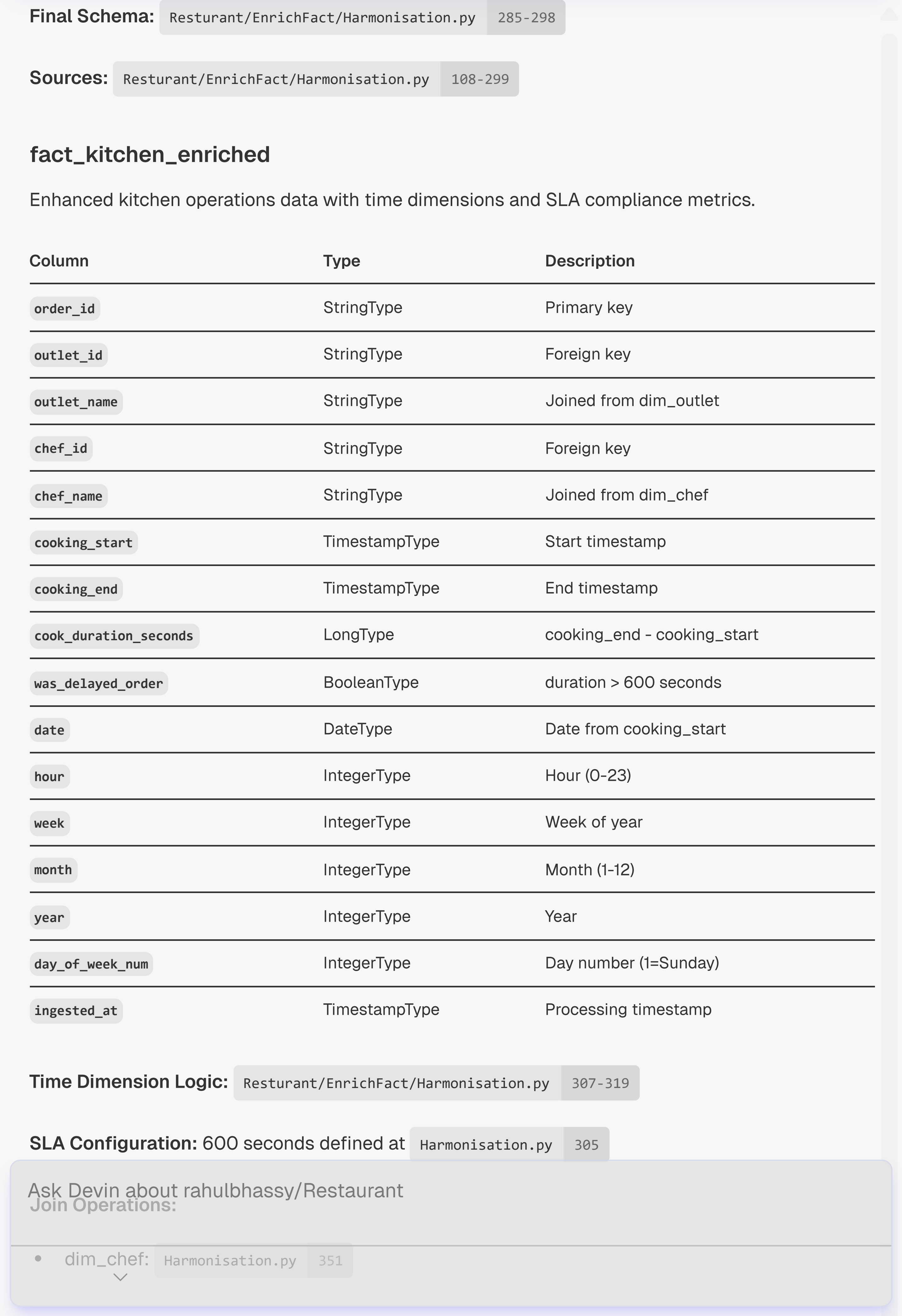
Column	Type	Logic
`avg_item_price`	DoubleType	total_order_price / total_items
`is_multi_item_order`	BooleanType	total_items > 1
`is_high_value_order`	BooleanType	total_order_price > 500
`is_digital_payment`	BooleanType	payment_mode in ("UPI", "Card", "Wallet")
`order_value_band`	StringType	"Low" (<200), "Medium" (200-500), "High" (>500)

Derivation Logic: `Resturant/EnrichFact/Harmonisation.py` 259-271

Base Attributes (Retained)

Ask Devin about rahulbhassy/Restaurant

Original columns from fact_sales: `order_id`, `order_ts`, `outlet_id`, `outlet_name` (joined), `customer_id`, `total_order_price`, `payment_mode`, `order_type`



Final Schema: `Resturant/EnrichFact/Harmonisation.py` 285-298

Sources: `Resturant/EnrichFact/Harmonisation.py` 108-299

fact_kitchen_enriched

Enhanced kitchen operations data with time dimensions and SLA compliance metrics.

Column	Type	Description
`order_id`	StringType	Primary key
`outlet_id`	StringType	Foreign key
`outlet_name`	StringType	Joined from dim_outlet
`chef_id`	StringType	Foreign key
`chef_name`	StringType	Joined from dim_chef
`cooking_start`	TimestampType	Start timestamp
`cooking_end`	TimestampType	End timestamp
`cook_duration_seconds`	LongType	cooking_end - cooking_start
`was_delayed_order`	BooleanType	duration > 600 seconds
`date`	DateType	Date from cooking_start
`hour`	IntegerType	Hour (0-23)
`week`	IntegerType	Week of year
`month`	IntegerType	Month (1-12)
`year`	IntegerType	Year
`day_of_week_num`	IntegerType	Day number (1=Sunday)
`ingested_at`	TimestampType	Processing timestamp

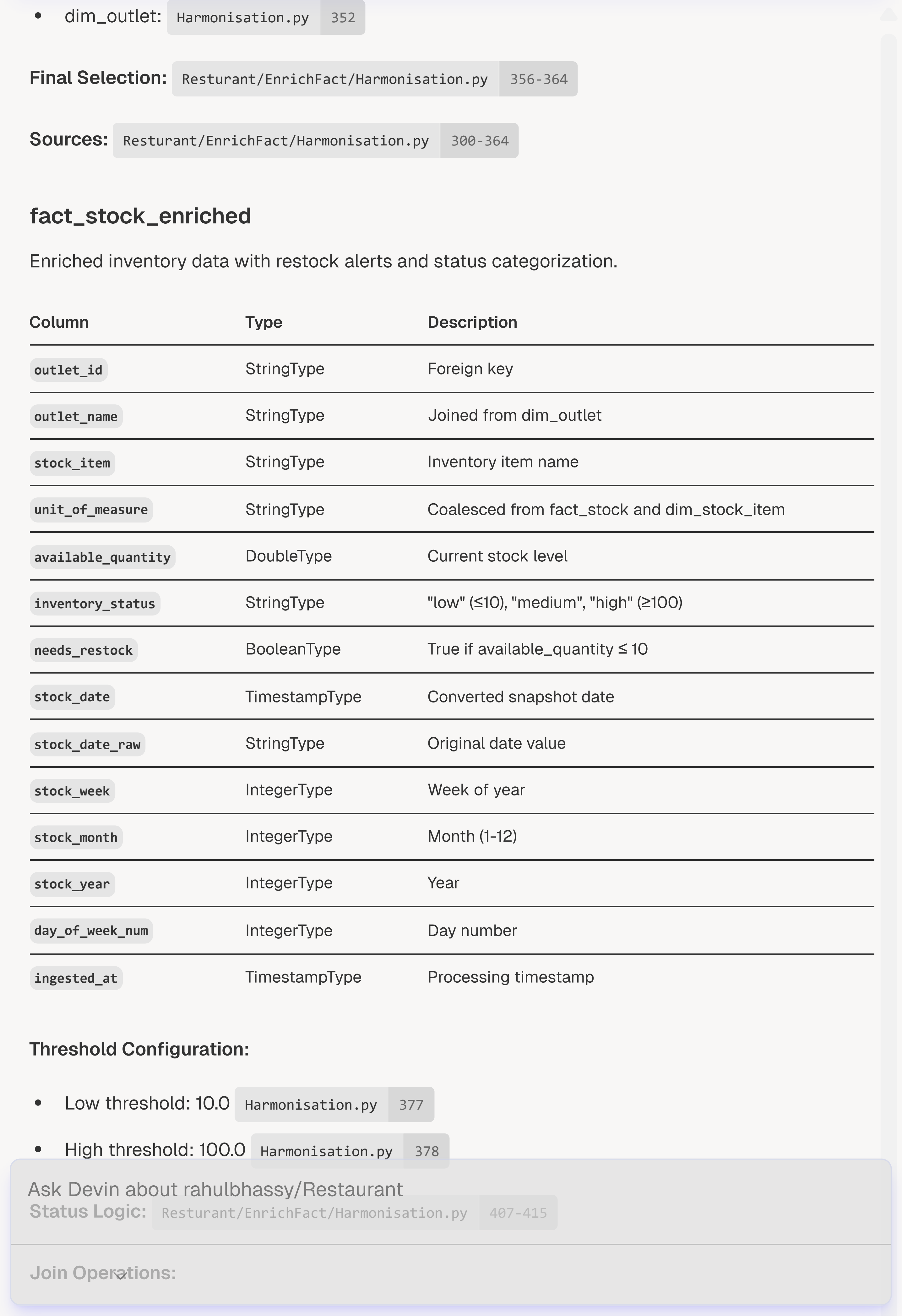
Time Dimension Logic: `Resturant/EnrichFact/Harmonisation.py` 307-319

SLA Configuration: 600 seconds defined at `Harmonisation.py` 305

Ask Devin about rahulbhassy/Restaurant

Join Operations:

- dim_chef: `Harmonisation.py` 351



- dim_outlet: `Harmonisation.py` 352

Final Selection: `Resturant/EnrichFact/Harmonisation.py` 356-364

Sources: `Resturant/EnrichFact/Harmonisation.py` 300-364

fact_stock_enriched

Enriched inventory data with restock alerts and status categorization.

Column	Type	Description
`outlet_id`	StringType	Foreign key
`outlet_name`	StringType	Joined from dim_outlet
`stock_item`	StringType	Inventory item name
`unit_of_measure`	StringType	Coalesced from fact_stock and dim_stock_item
`available_quantity`	DoubleType	Current stock level
`inventory_status`	StringType	"low" (≤10), "medium", "high" (≥100)
`needs_restock`	BooleanType	True if available_quantity ≤ 10
`stock_date`	TimestampType	Converted snapshot date
`stock_date_raw`	StringType	Original date value
`stock_week`	IntegerType	Week of year
`stock_month`	IntegerType	Month (1-12)
`stock_year`	IntegerType	Year
`day_of_week_num`	IntegerType	Day number
`ingested_at`	TimestampType	Processing timestamp

Threshold Configuration:

- Low threshold: 10.0 `Harmonisation.py` 377
- High threshold: 100.0 `Harmonisation.py` 378

Ask Devin about rahulbhassy/Restaurant

Status Logic: `Resturant/EnrichFact/Harmonisation.py` 407-415

Join Operations:

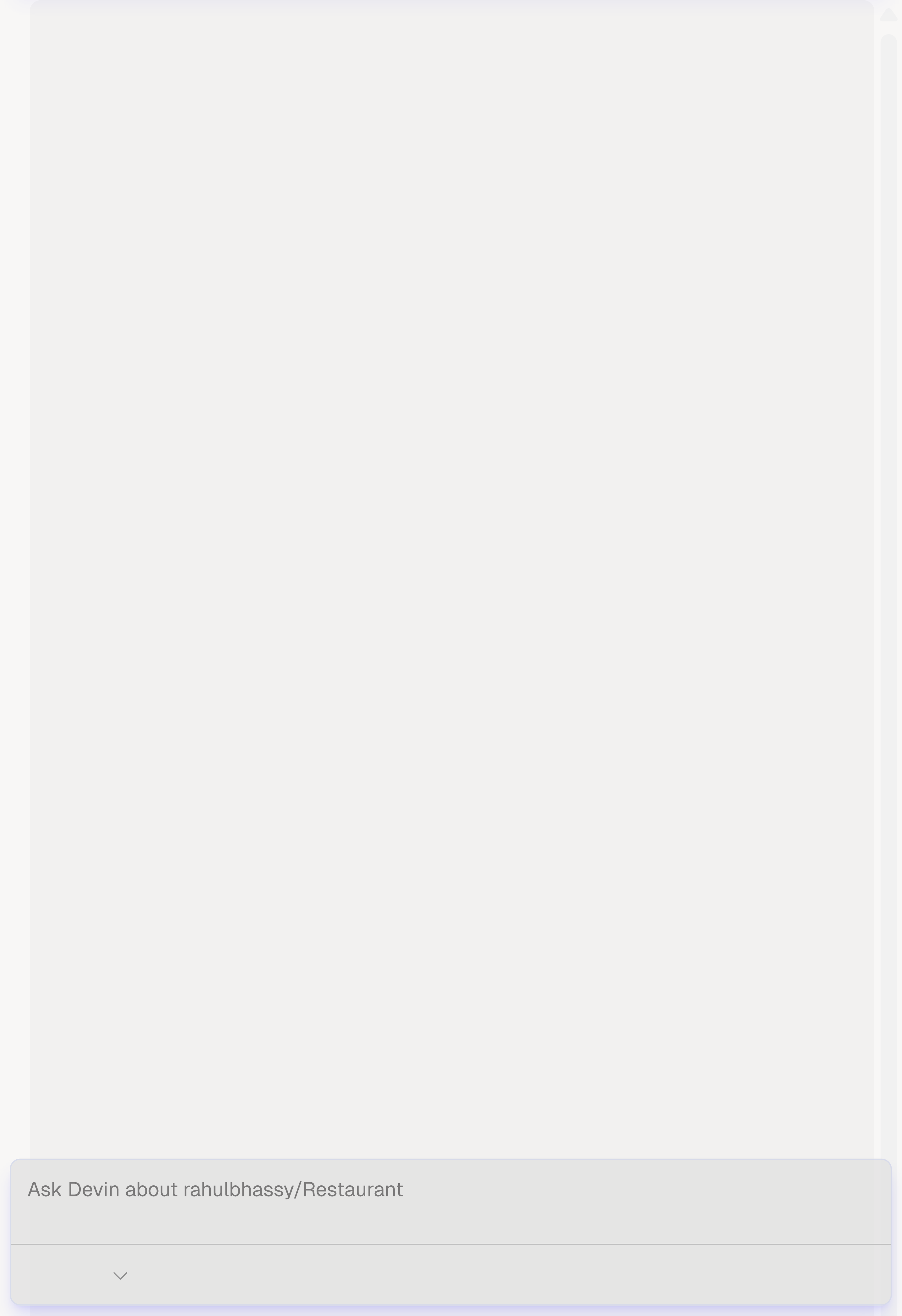
- dim_stock_item: Coalesce unit_of_measure `Harmonisation.py` `420-426`
- dim_outlet: `Harmonisation.py` `427`

Time Dimensions: `Resturant/EnrichFact/Harmonisation.py` `380-389`

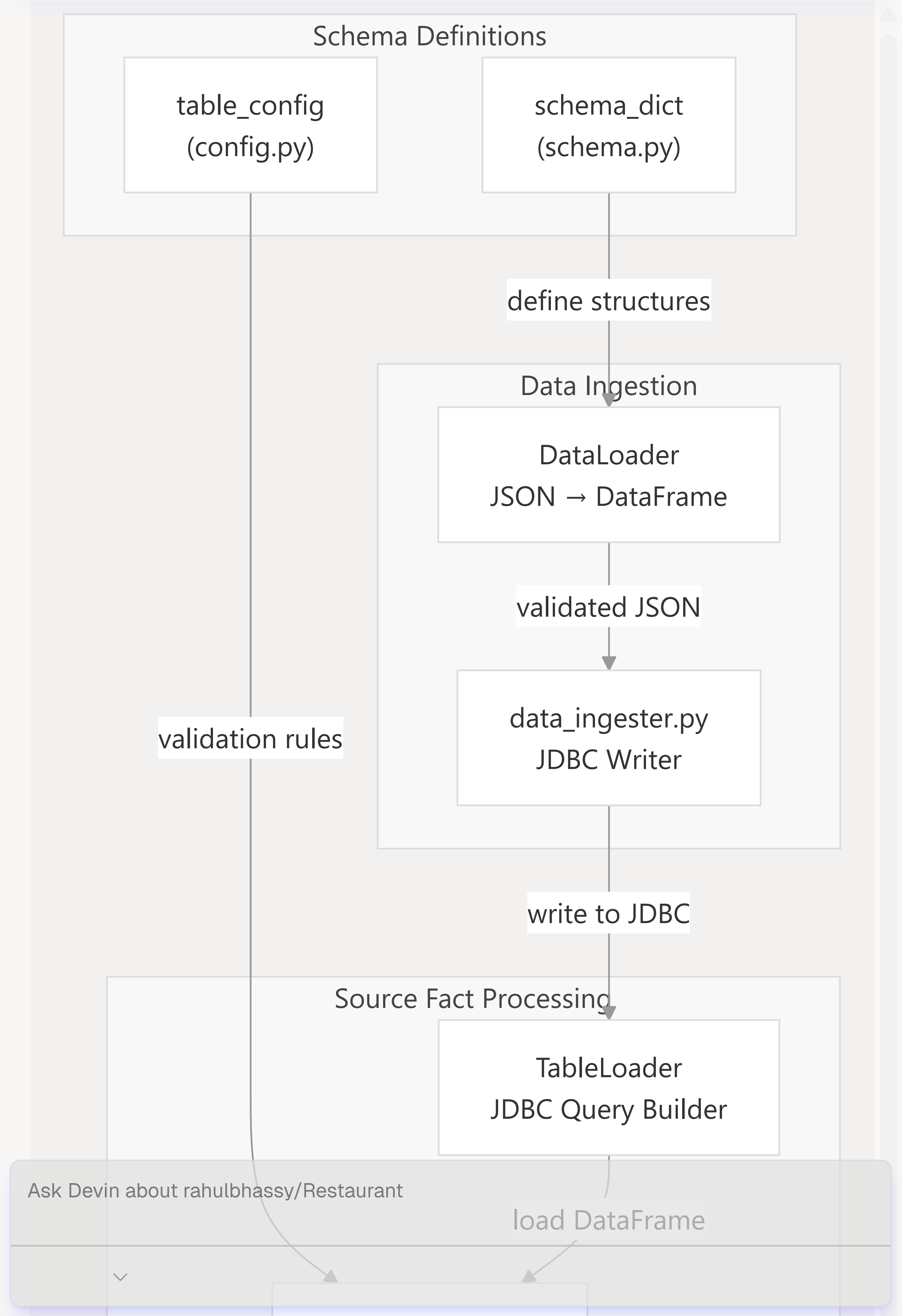
Sources: `Resturant/EnrichFact/Harmonisation.py` `372-442`

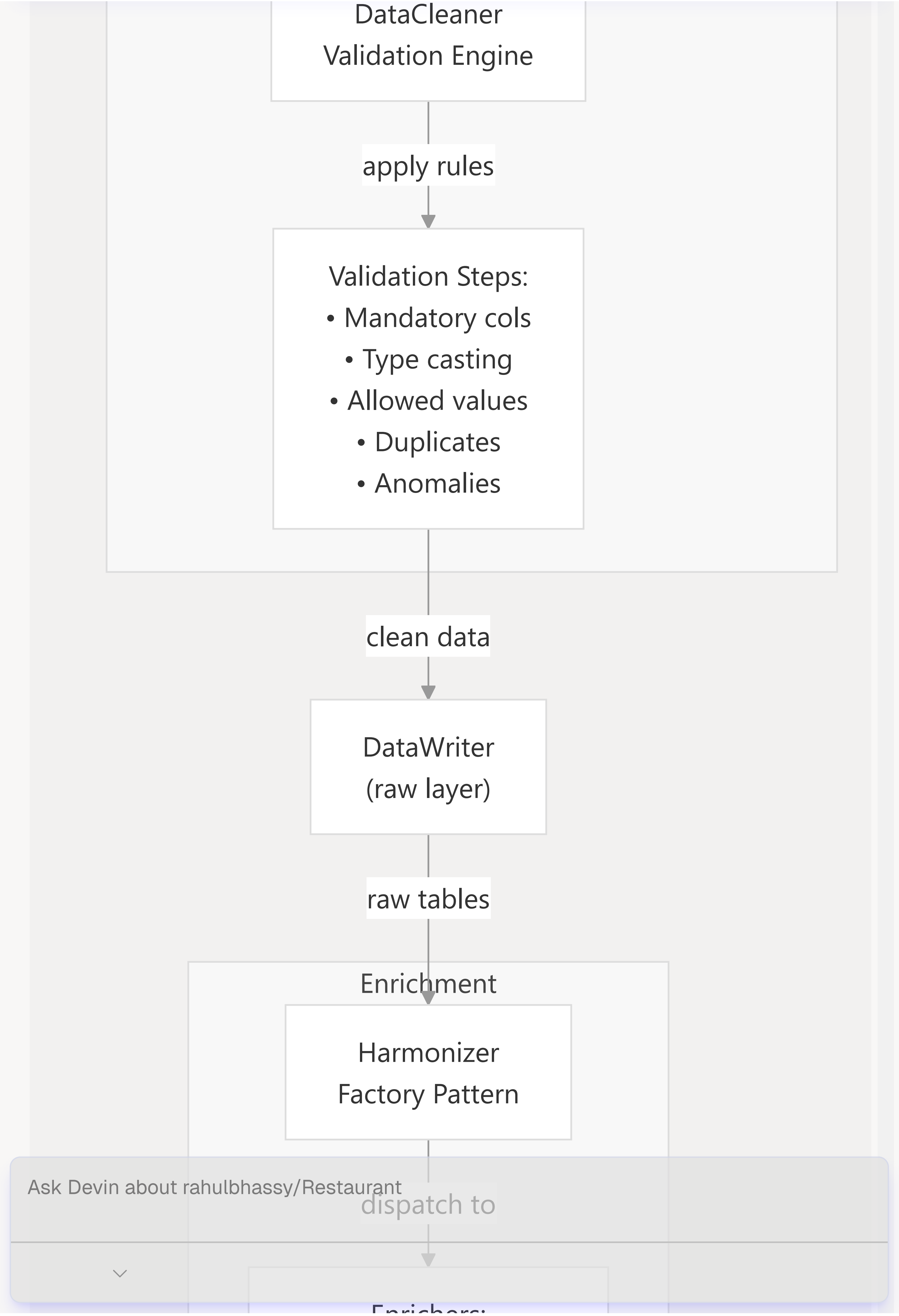
Schema Enforcement Architecture

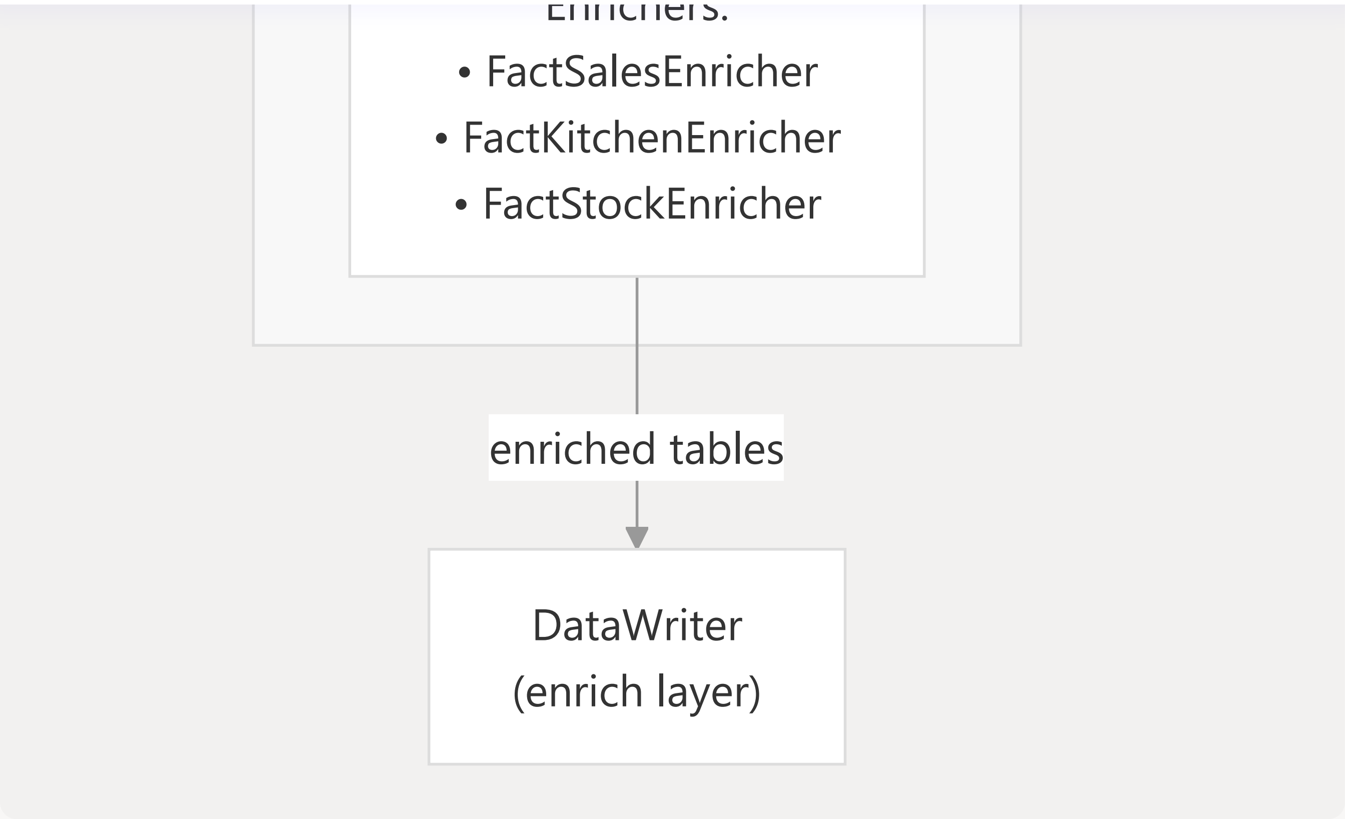
Ask Devin about rahulbhassy/Restaurant



Ask Devin about rahulbhassy/Restaurant


Schema Definitions
table_config
(config.py)
schema_dict
(schema.py)
define structures
Data Ingestion
DataLoader
JSON → DataFrame
validated JSON
data_ingester.py
JDBC Writer
validation rules
write to JDBC
Source Fact Processing
TableLoader
JDBC Query Builder
load DataFrame
Ask Devin about rahulbhassy/Restaurant


DataCleaner
Validation Engine
apply rules
Validation Steps:
• Mandatory cols
• Type casting
• Allowed values
• Duplicates
• Anomalies
clean data
DataWriter
(raw layer)
raw tables
Enrichment
Harmonizer
Factory Pattern
dispatch to
Enrichers:
Ask Devin about rahulbhassy/Restaurant



Validation Configuration

The `table_config` dictionary in `Resturant/SourceFact/config.py` `1-94` provides centralized validation rules:

Configuration Structure

```
table_config = {
    "table_name": {
        "mandatory_cols": [...]      # Required columns
        "cast_config": {...}         # Type casting map
        "allowed_values": {...}      # Enum validation
        "duplicate_keys": [...]      # Deduplication keys
        "anomaly_rules": {...}       # Threshold checks
    }
}
```

Validation Flow

The `DataCleaner` class applies validation in sequence:

1. **Mandatory Column Check:** Validates presence of required columns [Resturant/SourceFact/DataCleaner.py]
2. **Type Casting:** Converts columns using `cast_config` map
3. **Allowed Values:** Filters rows matching enum constraints
4. **Duplicate Removal:** Deduplicates based on `duplicate_keys`
5. **Anomaly Detection:** Flags/filters values exceeding thresholds

6. **Timestamp Addition:** Adds `ingested_at` column

Process Implementation: `Resturant/SourceFact/Process_SourceFact.py` 49-59

Delta Loading Configuration

The `delta_column_dict` specifies timestamp columns for incremental loading:

```
delta_column_dict = {
    "dim_customer": None,           # Full refresh only
    "dim_item": None,
    "dim_outlet": None,
    "dim_chef": None,
    "dim_stock_item": None,
    "fact_sales": "order_ts",       # Delta via order_ts
    "fact_kitchen": "cooking_end",  # Delta via cooking_end
    "fact_stock": None
}
```

Configuration: `Resturant/SourceFact/config.py` 79-88

Usage: `Resturant/SourceFact/Process_SourceFact.py` 21-39

Merge Key Configuration

The `merge_keys` dictionary defines primary keys for UPSERT operations during delta loads:

```
merge_keys = {
    "fact_sales": ["order_id"],
    "fact_kitchen": ["order_id"]
}
```

Configuration: `Resturant/SourceFact/config.py` 90-93

Usage: `Resturant/SourceFact/Process_SourceFact.py` 70-75

Sources: `Resturant/SourceFact/config.py` 1-94

`Resturant/SourceFact/Process_SourceFact.py` 1-79

Ask Devin about rahulbhassy/Restaurant

Schema Evolution and Versioning

Writer Mode Configuration

Dimension tables use `overwrite` mode while fact tables use `append` mode during initial data ingestion:

Ask Devin about rahulbhassy/Restaurant