Notebook 0 — Data Sanity & Setup

This notebook performs initial data sanity checks before analysis and modeling. We validate schema consistency, handle missing values, remove duplicates, and confirm that key identifiers (date, shop, item) align correctly. We also establish connections to BigQuery and confirm data availability at scale. This step ensures a reliable foundation for all downstream feature engineering and modeling.

In [1]:
```
# --- Cell 1: imports & runtime info ---
from pathlib import Path
import os, sys
import pandas as pd

from google.cloud import bigquery
from google.oauth2 import service_account

print("Python:", sys.executable)
print("CWD   :", Path.cwd())
```

```
Python: /home/btheard/retail-alpha-forecaster/.venv/bin/python
CWD   : /home/btheard/retail-alpha-forecaster/notebooks
```

In [2]:
```
# --- Cell 2: config (edit only if names differ) ---

PROJECT = "retail-alpha-forecaster"
DATASET = "raf"
FC_TABLE = f"`{PROJECT}.{DATASET}.forecasts`"

# Try common key locations; or rely on GOOGLE_APPLICATION_CREDENTIALS
CANDIDATES = [
    Path.cwd() / "keys" / "retail-alpha-forecaster-7f14a7b50e62.json",
    Path.cwd().parents[0] / "keys" / "retail-alpha-forecaster-7f14a7b50e62.j
]
KEY_PATH = next((p for p in CANDIDATES if p.exists()), None)

if KEY_PATH:
    creds = service_account.Credentials.from_service_account_file(str(KEY_PA
    client = bigquery.Client(project=PROJECT, credentials=creds, location="U
else:
    # Falls back to env or gcloud ADC
    client = bigquery.Client(project=PROJECT, location="US")

client.query("SELECT 1").result()
print("□ BigQuery client ready")
```

```
□ BigQuery client ready
```

In [3]:
```
# --- Cell 3: query helper ---
def q(sql: str) -> pd.DataFrame:
    job = client.query(sql)
    # Avoid BigQuery Storage API 403s
    return job.result().to_dataframe(create_bqstorage_client=False)
```

In [5]:
```
# --- Cell 4: health checks ---
meta = q(f"""
SELECT
  COUNT(*)                          AS n_rows,
  MIN(date)                         AS min_date,
  MAX(date)                         AS max_date,
  COUNTIF(shop_id IS NULL)          AS null_shop_rows,
  COUNTIF(item_id IS NULL)          AS null_item_rows
FROM {FC_TABLE}
""")
meta
```

Out[5]:

	n_rows	min_date	max_date	null_shop_rows	null_item_rows
0	25374	2015-10-01	2015-10-31	0	0

In [6]:
```
# --- Cell 5: coverage summary (non-null shop & item) ---
coverage = q(f"""
SELECT
  COUNT(*) AS n_rows,
  MIN(date) AS min_date,
  MAX(date) AS max_date,
  COUNT(DISTINCT shop_id) AS n_shops,
  COUNT(DISTINCT item_id) AS n_items,
  COUNT(DISTINCT CONCAT(CAST(shop_id AS STRING),':',CAST(item_id AS STRING))) AS n_pairs
FROM {FC_TABLE}
WHERE shop_id IS NOT NULL AND item_id IS NOT NULL
""")
coverage
```

Out[6]:

	n_rows	min_date	max_date	n_shops	n_items	n_pairs
0	25374	2015-10-01	2015-10-31	41	1374	4996

In [7]:
```
# --- Cell 6: pairs & windows (drives app dropdowns & date-pickers) ---
pairs = q(f"""
SELECT shop_id, item_id, COUNT(*) AS n_rows
FROM {FC_TABLE}
WHERE shop_id IS NOT NULL AND item_id IS NOT NULL
GROUP BY shop_id, item_id
ORDER BY n_rows DESC
""")
pairs.head(20)
```

Out[7]:

	shop_id	item_id	n_rows
0	25	20949	62
1	31	20949	62
2	42	20949	62
3	26	20949	60
4	28	20949	60
5	47	20949	60
6	57	20949	60
7	21	20949	58
8	39	20949	58
9	55	13097	58
10	55	13099	56
11	56	20949	56
12	14	20949	54
13	55	492	54
14	58	20949	54
15	4	20949	52
16	6	20949	52
17	38	20949	52
18	16	20949	52
19	7	20949	50

In [8]:

```
# --- Cell 6b: min/max window per pair ---
pair_windows = q(f"""
SELECT
  shop_id, item_id,
  MIN(date) AS min_date,
  MAX(date) AS max_date,
  COUNT(*)  AS n_rows
FROM {FC_TABLE}
WHERE shop_id IS NOT NULL AND item_id IS NOT NULL
GROUP BY shop_id, item_id
ORDER BY n_rows DESC
""")
pair_windows.head(20)
```

Out[8]:

	shop_id	item_id	min_date	max_date	n_rows
0	25	20949	2015-10-01	2015-10-31	62
1	31	20949	2015-10-01	2015-10-31	62
2	42	20949	2015-10-01	2015-10-31	62
3	26	20949	2015-10-01	2015-10-31	60
4	28	20949	2015-10-01	2015-10-31	60
5	47	20949	2015-10-01	2015-10-31	60
6	57	20949	2015-10-01	2015-10-31	60
7	21	20949	2015-10-01	2015-10-31	58
8	39	20949	2015-10-01	2015-10-31	58
9	55	13097	2015-10-02	2015-10-31	58
10	55	13099	2015-10-01	2015-10-31	56
11	56	20949	2015-10-01	2015-10-31	56
12	14	20949	2015-10-01	2015-10-31	54
13	55	492	2015-10-01	2015-10-31	54
14	58	20949	2015-10-01	2015-10-31	54
15	4	20949	2015-10-01	2015-10-29	52
16	6	20949	2015-10-01	2015-10-31	52
17	38	20949	2015-10-01	2015-10-31	52
18	16	20949	2015-10-02	2015-10-31	52
19	7	20949	2015-10-01	2015-10-31	50

In [9]:
```
# --- Cell 7: sanity checks on yhat & bands ---
sanity = q(f"""
SELECT
  COUNTIF(yhat IS NULL)            AS null_yhat,
  COUNTIF(yhat < 0)                AS negative_yhat,
  COUNTIF(yhat_lower > yhat_upper) AS swapped_bands,
  COUNT(DISTINCT DATE(date))       AS n_days
FROM {FC_TABLE}
WHERE shop_id IS NOT NULL AND item_id IS NOT NULL
""")
sanity
```

Out[9]:

	null_yhat	negative_yhat	swapped_bands	n_days
0	0	4	0	31

In [10]:
```
# --- Cell 7b: check duplicate keys (date, shop_id, item_id) ---
dupes = q(f"""
SELECT date, shop_id, item_id, COUNT(*) AS cnt
```

```
FROM {FC_TABLE}
WHERE shop_id IS NOT NULL AND item_id IS NOT NULL
GROUP BY date, shop_id, item_id
HAVING COUNT(*) > 1
ORDER BY cnt DESC, date
""")
dupes.head(20), len(dupes)
```

Out[10]:

```
(          date  shop_id  item_id  cnt
 0   2015-10-01        2     7894    2
 1   2015-10-01        2    17717    2
 2   2015-10-01        3     5671    2
 3   2015-10-01        3     6738    2
 4   2015-10-01        3     6740    2
 5   2015-10-01        3    17717    2
 6   2015-10-01        4     3731    2
 7   2015-10-01        4     7736    2
 8   2015-10-01        4    17717    2
 9   2015-10-01        4    20949    2
 10  2015-10-01        5     3733    2
 11  2015-10-01        5     4351    2
 12  2015-10-01        5     7018    2
 13  2015-10-01        5     7791    2
 14  2015-10-01        5    14227    2
 15  2015-10-01        5    15045    2
 16  2015-10-01        5    16287    2
 17  2015-10-01        5    20949    2
 18  2015-10-01        6       31    2
 19  2015-10-01        6     4181    2,
 12687)
```

In [11]:

```
# --- Cell 8: create a clean view in BigQuery (idempotent) ---

sql_clean = f"""
CREATE OR REPLACE VIEW `{PROJECT}.{DATASET}.v_forecasts_clean` AS
SELECT
  date,
  shop_id,
  item_id,
  model,
  yhat,
  LEAST(yhat_lower, yhat_upper) AS yhat_lower,
  GREATEST(yhat_lower, yhat_upper) AS yhat_upper,
  created_at
FROM (
  SELECT *,
    ROW_NUMBER() OVER (
      PARTITION BY date, shop_id, item_id
      ORDER BY
        CASE WHEN model = 'lightgbm' THEN 0 ELSE 1 END,
        created_at DESC
    ) AS rn
  FROM {FC_TABLE}
  WHERE shop_id IS NOT NULL AND item_id IS NOT NULL
)
WHERE rn = 1
```

```
"""
client.query(sql_clean).result()
print("✅ Created/updated view raf.v_forecasts_clean")
```

```
✅ Created/updated view raf.v_forecasts_clean
```

In [12]:
```
# --- Cell 9: pair & list helper views (for Streamlit dropdowns quickly) ---

sql_pairs = f"""
CREATE OR REPLACE VIEW `{PROJECT}.{DATASET}.v_forecast_pairs` AS
SELECT
  shop_id, item_id,
  MIN(date) AS min_date,
  MAX(date) AS max_date,
  COUNT(*)  AS n_rows
FROM `{PROJECT}.{DATASET}.v_forecasts_clean`
GROUP BY shop_id, item_id
"""
client.query(sql_pairs).result()

sql_lists = f"""
CREATE OR REPLACE VIEW `{PROJECT}.{DATASET}.v_shops_items` AS
WITH s AS (SELECT DISTINCT shop_id FROM `{PROJECT}.{DATASET}.v_forecasts_cle
     i AS (SELECT DISTINCT item_id FROM `{PROJECT}.{DATASET}.v_forecasts_cle
SELECT * FROM s, i
"""
client.query(sql_lists).result()

print("✅ Created/updated views raf.v_forecast_pairs, raf.v_shops_items")
```

```
✅ Created/updated views raf.v_forecast_pairs, raf.v_shops_items
```

In [13]:
```
# --- Cell 10: preview one pair (edit IDs and re-run) ---
SHOP, ITEM = tuple(pairs[['shop_id','item_id']].iloc[0])  # top pair by rows
preview = q(f"""
SELECT date, yhat, yhat_lower, yhat_upper, model, created_at
FROM `{PROJECT}.{DATASET}.v_forecasts_clean`
WHERE shop_id = {SHOP} AND item_id = {ITEM}
ORDER BY date
""")
preview.head(20)
```

Out[13]:

	date	yhat	yhat_lower	yhat_upper	model	created_at
0	2015-10-01	11.329830	7.728220	14.931440	lightgbm	2025-08-17 16:15:26.225742+00:00
1	2015-10-02	15.140896	11.539286	18.742506	lightgbm	2025-08-17 16:15:26.225742+00:00
2	2015-10-03	13.465129	9.863519	17.066739	lightgbm	2025-08-17 16:15:26.225742+00:00
3	2015-10-04	15.485568	11.883958	19.087178	lightgbm	2025-08-17 16:15:26.225742+00:00
4	2015-10-05	10.192873	6.591263	13.794483	lightgbm	2025-08-17 16:15:26.225742+00:00
5	2015-10-06	12.013605	8.411995	15.615215	lightgbm	2025-08-17 16:15:26.225742+00:00
6	2015-10-07	9.743950	6.142340	13.345560	lightgbm	2025-08-17 16:15:26.225742+00:00
7	2015-10-08	10.896068	7.294458	14.497678	lightgbm	2025-08-17 16:15:26.225742+00:00
8	2015-10-09	14.232312	10.630702	17.833922	lightgbm	2025-08-17 16:15:26.225742+00:00
9	2015-10-10	21.342919	17.741309	24.944529	lightgbm	2025-08-17 16:15:26.225742+00:00
10	2015-10-11	14.108944	10.507334	17.710554	lightgbm	2025-08-17 16:15:26.225742+00:00
11	2015-10-12	9.901458	6.299848	13.503068	lightgbm	2025-08-17 16:15:26.225742+00:00
12	2015-10-13	12.392951	8.791341	15.994561	lightgbm	2025-08-17 16:15:26.225742+00:00
13	2015-10-14	13.013358	9.411748	16.614968	lightgbm	2025-08-17 16:15:26.225742+00:00
14	2015-10-15	11.451318	7.849708	15.052929	lightgbm	2025-08-17 16:15:26.225742+00:00
15	2015-10-16	15.932700	12.331090	19.534310	lightgbm	2025-08-17 16:15:26.225742+00:00
16	2015-10-17	19.483808	15.882198	23.085418	lightgbm	2025-08-17 16:15:26.225742+00:00
17	2015-10-18	14.153442	10.551832	17.755052	lightgbm	2025-08-17 16:15:26.225742+00:00
18	2015-10-19	9.636790	6.035179	13.238400	lightgbm	2025-08-17 16:15:26.225742+00:00
19	2015-10-20	14.211178	10.609568	17.812789	lightgbm	2025-08-17 16:15:26.225742+00:00

Takeaways

- **Non-null, deduped forecasts** now available at: `raf.v_forecasts_clean`.
- **Valid shop–item pairs + date windows** at: `raf.v_forecast_pairs`.
- These views drive the Streamlit dashboard dropdowns & date pickers.
- If a pair shows only a single date, the app will display a single point; choose a pair with more rows for a richer view.