New Technologies in Informatics

THE DATA ANALYSIS OF CAR SALES

Made by: Francisco Faller, Angel Ocariz, Alejandro Ponce, Derek Rodriguez and Diego González

WHAT IS THE MAIN GOAL?

Problem

Automakers need to **understand historical sales**, **price dynamics**, and advertising pressure to **plan inventory** and marketing.

Goal

Build an ITD pipeline that unifies models, sales, prices, trims, and ads, then delivers views and a dashboard that explain YoY growth, price gaps, and highly advertised models.

SOURCES AND DATASET .

Car Sales is a unified dataset for analyzing the automotive market.

Main tables

CarModel: Basic information of car models.

Trim: Details about specific versions (trims) of each model.

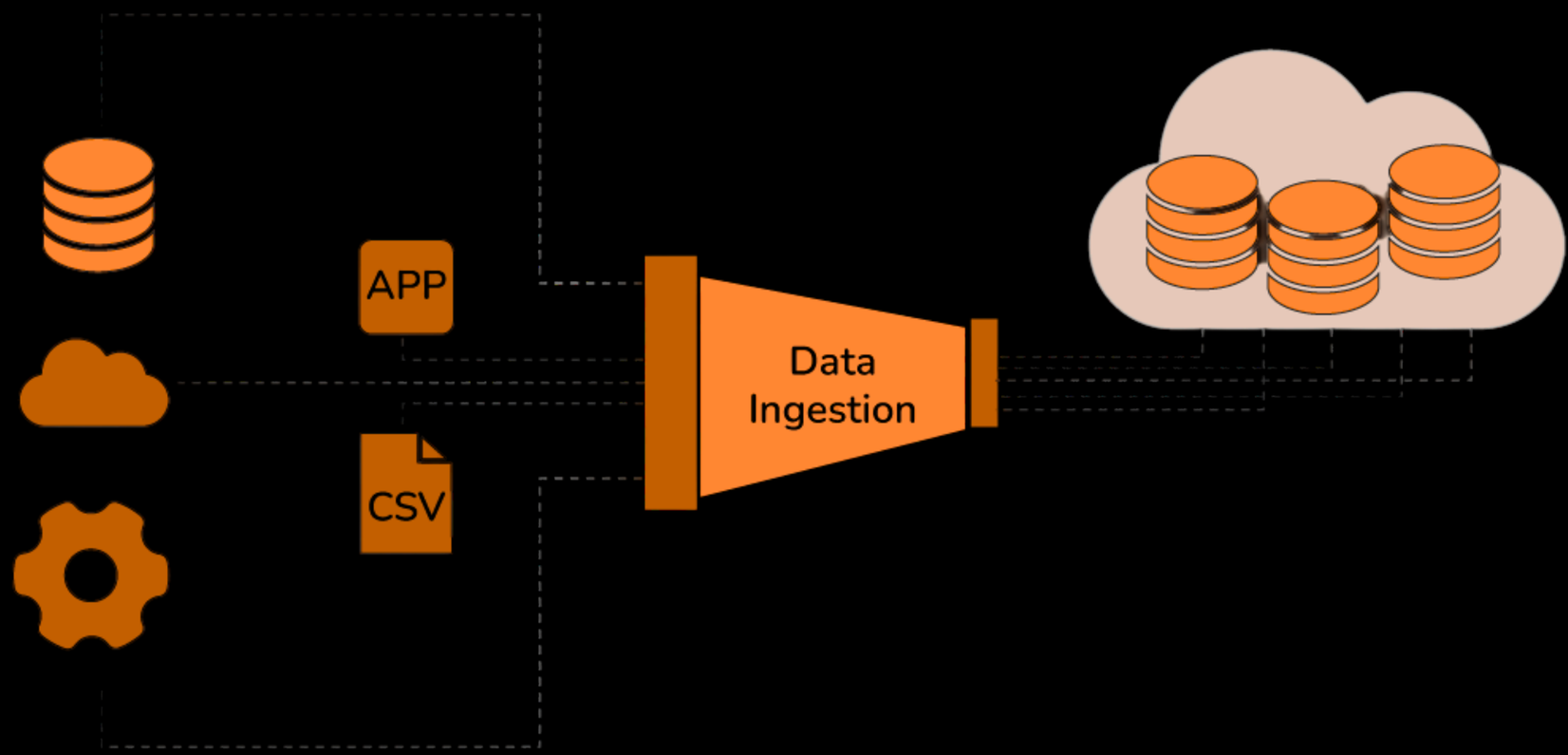
Price: Entry price per year for each car model.

Image: Links car models with their related images.

Sales: Historical car sales by year (2001–2020).

Advertisement: Data from car sale advertisements or used car listings.

INGESTION



DATABASE STRUCTURE

Raw schema

Landing zone for source files exactly as they arrive from S3, no cleaning or reshaping. It preserves the original record for reprocessing and audits. Holds CarModel, Trim, Price, Sales, Advertisement, and Image.

Tables created in this schema:

- CARMODEL
- TRIM
- PRICE
- SALES
- ADVERTISEMENT
- IMAGE

Harmonized Schema

Standardized, typed data ready for reliable joins. We cast numbers safely, normalize maker and model names, and unpivot Sales to Year and Units_sold. Includes joins like CarModel with Trim to create cardata.

Tables created in this schema:

- cardata (CarModel + Trim)
- carprice (CarModel + Price)
- carsales (Sales unpivoted to long)
- caradvertisement (clean ads with specs)
- carimage (image catalog)

DATABASE STRUCTURE

Analytics schema

Business facing model with facts, dimensions, and ready to use views. Facts cover sales, prices, and ads. Dimensions describe cars, dates, and images. Views power the dashboards, like YoY growth and price gap.

- Facts: fact_sales, fact_price, fact_advertisement
- Dimensions: dim_car, dim_date, dim_image
- Views: v_top10_cars_by_sales, v_avg_price_maker, v_entry_vs_adv_price, v_sales_growth_maker, v_top10_most_advs_vs_most_sold

Automation Schema

Operational brain of the pipeline. Stored procedures and tasks load new files, run transforms in the right order, and rebuild the analytics layer so the dashboard stays current without manual steps.

- Procedures: sp_load_sales, sp_transform_cardata, sp_transform_carprice, sp_transform_carsales, sp_transform_caradvertisement, sp_transform_carimage, sp_transform_all, sp_build_fact and dim tables, sp_build_analytics_all
- Tasks: task_load_sales, task_transform_all, task_build_analytics

DATABASE STRUCTURE

Public Schema

Ingestion utilities only. Defines the S3 storage integration, CSV file format, and the stage that points to the bucket. It enables secure, repeatable movement of data into raw.

```
CREATE OR REPLACE STORAGE INTEGRATION CARSFINAL_S3_role_integration
    TYPE = EXTERNAL_STAGE
    STORAGE_PROVIDER = S3
    ENABLED = TRUE
    STORAGE_AWS_ROLE_ARN = 'arn:aws:iam::650155822305:role/snowflake_role'
    STORAGE_ALLOWED_LOCATIONS = ('s3://balatros-cars-data-set');
```

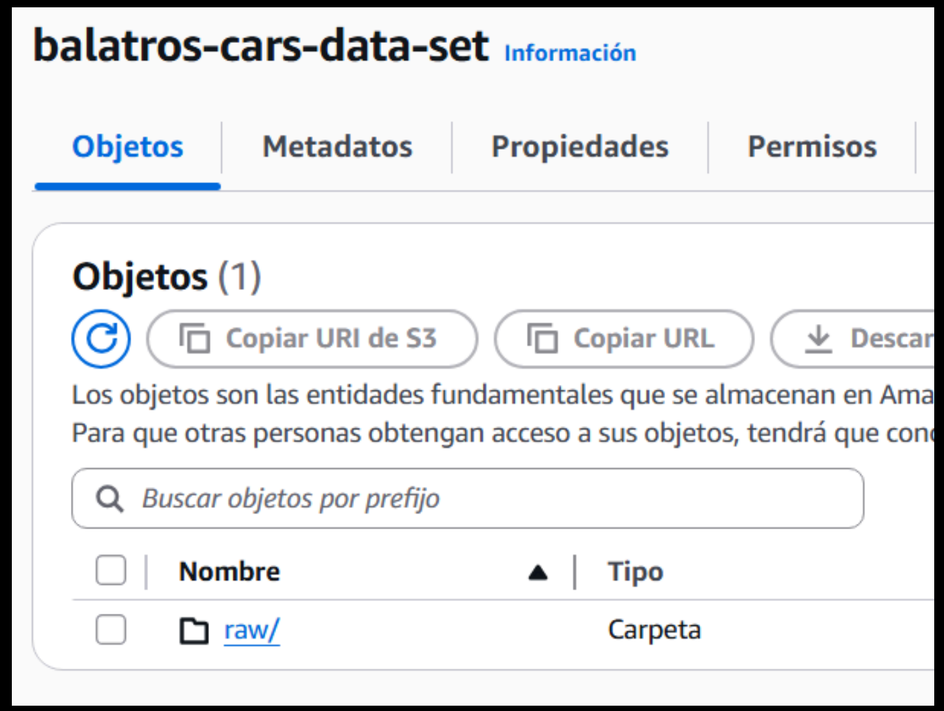
```
CREATE OR REPLACE STAGE public.s3load_stage
    URL = 's3://balatros-cars-data-set'
    STORAGE_INTEGRATION = CARSFINAL_S3_role_integration
    FILE_FORMAT = public.csv_ff;
```

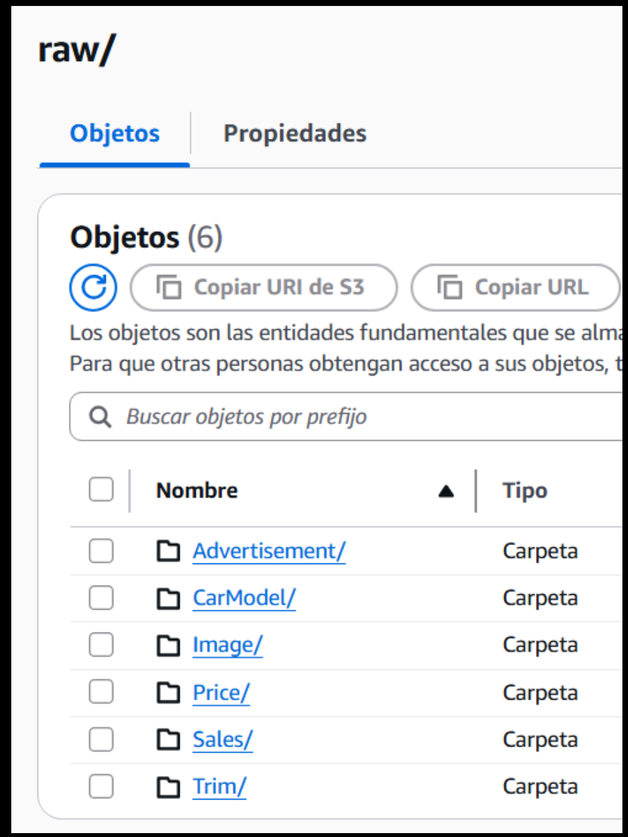
```
CREATE OR REPLACE FILE FORMAT public.csv_ff
    TYPE = 'csv'
    SKIP_HEADER = 1
    FIELD_OPTIONALLY_ENCLOSED_BY ='"'
    TRIM_SPACE = TRUE
    NULL_IF = ('NULL');
```

AWS

S3 Bucket

We set up an Amazon S3 bucket, where each table is stored as a CSV in its own subfolder under the raw directory for Snowflake ingestion.





AWS

Storage Integration

We defined a secure connection in Snowflake to access S3, where the function specifies the storage provider, allowed bucket locations, and links to the AWS IAM role created.

```
USE SCHEMA public;

CREATE OR REPLACE STORAGE INTEGRATION CARSFINAL_S3_role_integration
    TYPE = EXTERNAL_STAGE
    STORAGE_PROVIDER = S3
    ENABLED = TRUE
    STORAGE_AWS_ROLE_ARN = 'arn:aws:iam::650155822305:role/snowflake_role'
    STORAGE_ALLOWED_LOCATIONS = ('s3://balatros-cars-data-set');

SHOW INTEGRATIONS;

DESCRIBE INTEGRATION CARSFINAL_S3_ROLE_INTEGRATION;
```

Entidades de confianza

Entidades que pueden asumir este rol en condiciones especificadas.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::921821544361:user/t8j41000-s"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "sts:ExternalId": [
                        "GGB82381_SFCRole=700_Ssrzkga+vzzqqP7ogb36Xp393Ag=",
                        "GGB82381_SFCRole=1570_01TIOwi3T6GjoXshfLe+jFkeb6U="
                    ]
                }
            }
        }
    ]
}
```

IAM Role

By using the Describe Integration, the IAM Role ARN and External IDs that Snowflake generates, which we then added to the AWS IAM trust policy so Snowflake can assume the role.

TABLES IN RAW SCHEMA

Raw data ingested directly from S3, without cleaning or type conversions

```sql
-- CarModel
CREATE TABLE CARSALESFINAL.raw.CARMODEL (
    Automaker VARCHAR,
    Automaker_ID VARCHAR,
    Genmodel VARCHAR,
    Genmodel_ID VARCHAR
);
```

Master catalog of brands and models.

```sql
-- Trim
CREATE TABLE CARSALESFINAL.raw.TRIM (
    Genmodel_ID VARCHAR,
    Maker VARCHAR,
    Genmodel VARCHAR,
    Trim VARCHAR,
    Year VARCHAR,
    Price VARCHAR,
    Gas_emission VARCHAR,
    Fuel_type VARCHAR,
    Engine_size VARCHAR
);
```

Versions/trims by model and year (engine, fuel, emissions, raw price). Technical specs of each car.

```sql
-- Sales (wide table with one column per year)
CREATE TABLE CARSALESFINAL.raw.SALES (
    Maker VARCHAR,
    Genmodel VARCHAR,
    Genmodel_ID VARCHAR,
    "2020" VARCHAR,
    "2019" VARCHAR,
    "2018" VARCHAR,
    "2017" VARCHAR,
    "2016" VARCHAR,
    "2015" VARCHAR,
    "2014" VARCHAR,
    "2013" VARCHAR,
    "2012" VARCHAR,
    "2011" VARCHAR,
    "2010" VARCHAR,
    "2009" VARCHAR,
    "2008" VARCHAR,
    "2007" VARCHAR,
    "2006" VARCHAR,
    "2005" VARCHAR,
    "2004" VARCHAR,
    "2003" VARCHAR,
    "2002" VARCHAR,
    "2001" VARCHAR
);
```

Historical sales by model .

```sql
-- Advertisement
CREATE OR REPLACE TABLE CARSALESFINAL.raw.ADVERTISEMENT (
    Maker VARCHAR,
    Genmodel VARCHAR,
    Genmodel_ID VARCHAR,
    Adv_ID VARCHAR,
    Adv_year VARCHAR,
    Adv_month VARCHAR,
    Color VARCHAR,
    Reg_year VARCHAR,
    Bodytype VARCHAR,
    Runned_Miles VARCHAR,
    Engin_size VARCHAR,
    Gearbox VARCHAR,
    Fuel_type VARCHAR,
    Price VARCHAR,
    Engine_power VARCHAR,
    Annual_Tax VARCHAR,
    Wheelbase VARCHAR,
    Height VARCHAR,
    Width VARCHAR,
    Length VARCHAR,
    Average_mpg VARCHAR,
    Top_speed VARCHAR,
    Seat_num VARCHAR,
    Door_num VARCHAR
);
```

ad year/month plus attributes (price, mileage, engine, gearbox, doors, etc.).

```sql
-- Price
CREATE TABLE CARSALESFINAL.raw.PRICE (
    Maker VARCHAR,
    Genmodel VARCHAR,
    Genmodel_ID VARCHAR,
    Year VARCHAR,
    Entry_price VARCHAR
);
```

Entry price by Genmodel_ID and year. Used to compare against advertisement prices.

```sql
-- Image
CREATE TABLE CARSALESFINAL.raw.IMAGE (
    Genmodel_ID VARCHAR,
    Image_ID VARCHAR,
    Image_name VARCHAR,
    Predicted_viewpoint VARCHAR,
    Quality_check VARCHAR
);
```

Metadata of images per Genmodel_ID (file name, viewpoint, quality flag).

COPYING DATA

Manual ingestion of data using COPY INTO

```
COPY INTO CARSALESFINAL.raw.CARMODEL
FROM @public.s3load_stage/raw/CarModel;

COPY INTO CARSALESFINAL.raw.TRIM
FROM @public.s3load_stage/raw/Trim;

COPY INTO CARSALESFINAL.raw.PRICE
FROM @public.s3load_stage/raw/Price;

COPY INTO CARSALESFINAL.raw.IMAGE
FROM @public.s3load_stage/raw/Image;

COPY INTO CARSALESFINAL.raw.ADVERTISEMENT
FROM @public.s3load_stage/raw/Advertisement;
```

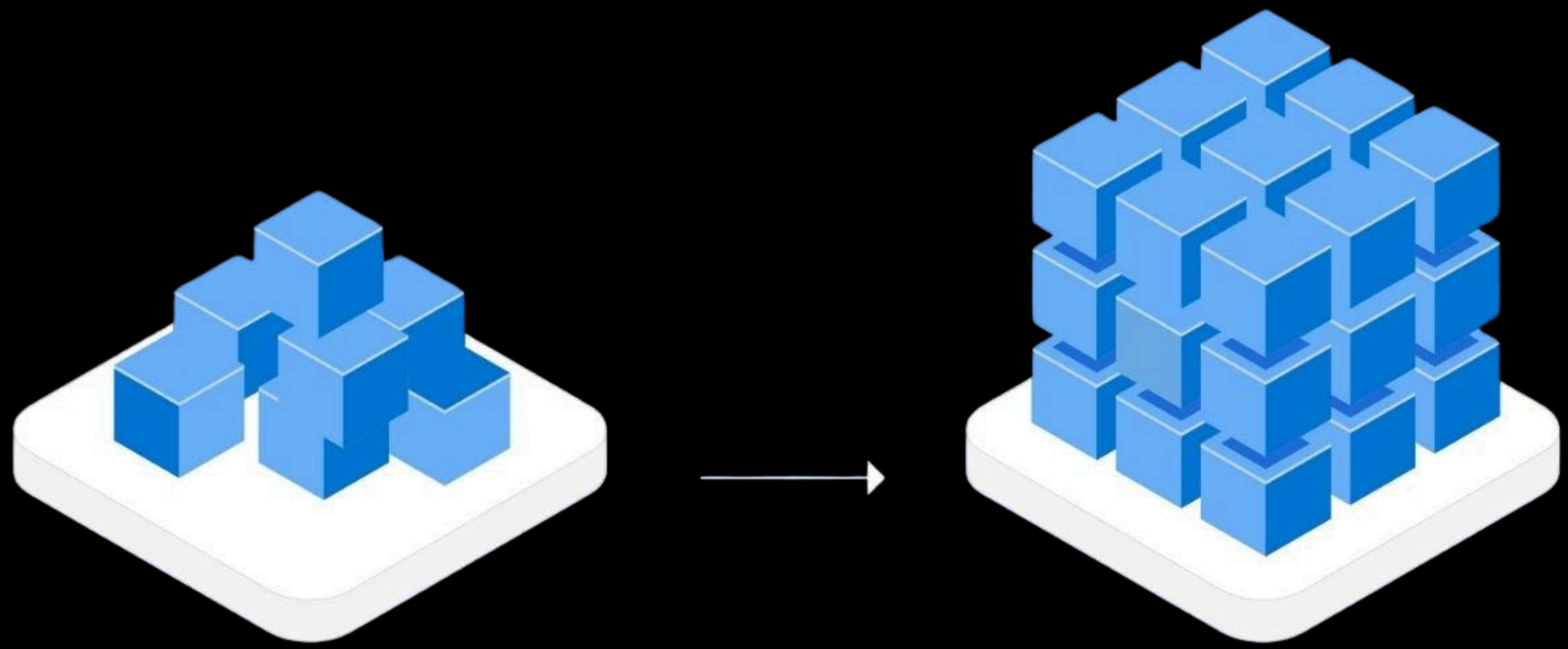
Automate the ingestion by using a procedure

```
USE SCHEMA automation;

CREATE OR REPLACE PROCEDURE sp_load_sales()
    RETURNS STRING
    LANGUAGE SQL
AS
$$
BEGIN
    COPY INTO CARSALESFINAL.raw.SALES
    FROM @public.s3load_stage/raw/Sales/
    FILE_FORMAT = (FORMAT_NAME = 'public.csv_ff')
    ON_ERROR = 'CONTINUE';

    RETURN 'Sales load completed';
END;
$$;
```

Creating the task to automatize the procedure

```
CREATE OR REPLACE TASK task_load_sales
    WAREHOUSE = CARSALESFINAL_WH
    SCHEDULE = 'USING CRON */2 * * * * UTC'
AS
    CALL sp_load_sales();
```

TRANSFORMATION



HARMONIZED TABLES

These are the tables created utilizing the harmonized schema to standardize and clean the raw data, ensuring a consistent structure.

```
CREATE OR REPLACE TABLE CARSALESFINAL.harmonized.cardata (
    Genmodel_ID VARCHAR,
    Automaker VARCHAR,
    Automaker_ID VARCHAR,
    Genmodel VARCHAR,
    Trim VARCHAR,
    Year INT,
    Price NUMBER(12,2),
    Gas_emission NUMBER(10,2),
    Fuel_type VARCHAR,
    Engine_size NUMBER(10,2)
);
```

Combines CarModel + Trim.
Cleans attributes like year, price, engine size, and emissions.

```
CREATE OR REPLACE TABLE CARSALESFINAL.harmonized.carprice (
    Genmodel_ID VARCHAR,
    Automaker VARCHAR,
    Genmodel VARCHAR,
    Year INT,
    Entry_price NUMBER(12,2)
);
```

Combines CarModel + Price.
Entry prices cast to numeric values for each year.

```
CREATE OR REPLACE TABLE CARSALESFINAL.harmonized.carsales (
    Genmodel_ID VARCHAR,
    Genmodel VARCHAR,
    Maker VARCHAR,
    Year INT,
    Units_sold INT
);
```

Converts the wide sales table (2001–2020 as columns) into a long format: year → units sold.

```
CREATE OR REPLACE TABLE CARSALESFINAL.harmonized.caradvertisement (
    Adv_ID VARCHAR,
    Genmodel_ID VARCHAR,
    Genmodel VARCHAR,
    Maker VARCHAR,
    Adv_year INT,
    Adv_month INT,
    Reg_year INT,
    Bodytype VARCHAR,
    Color VARCHAR,
    Runned_Miles INT,
    Engin_size NUMBER(10,2),
    Gearbox VARCHAR,
    Fuel_type VARCHAR,
    Price NUMBER(12,2),
    Engine_power NUMBER(10,2),
    Annual_Tax NUMBER(10,2),
    Wheelbase NUMBER(10,2),
    Height NUMBER(10,2),
    Width NUMBER(10,2),
    Length NUMBER(10,2),
    Average_mpg NUMBER(10,2),
    Top_speed NUMBER(10,2),
    Seat_num SMALLINT,
    Door_num SMALLINT
);
```

Transforms raw ad listings into clean structured records with correct numeric types (mileage, tax, engine power, dimensions, etc.).

```
CREATE OR REPLACE TABLE CARSALESFINAL.harmonized.carimage (
    Genmodel_ID VARCHAR,
    Genmodel VARCHAR,
    Image_ID VARCHAR,
    Image_name VARCHAR,
    Predicted_viewpoint VARCHAR,
    Quality_check VARCHAR
);
```

Joins image metadata with CarModel to provide context per image.

DATA TRANSFORMATION PROCEDURES

The procedure transform cardata combined model and trim information, as well as cleaned and standardized details for consistency.

```
-- 1. Transform CARDATA (CarModel + Trim)
CREATE OR REPLACE PROCEDURE sp_transform_cardata()
RETURNS STRING
LANGUAGE SQL
AS
$$
BEGIN
  INSERT OVERWRITE INTO CARSALESFINAL.harmonized.cardata
  SELECT
    t.Genmodel_ID,
    c.Automaker,
    c.Automaker_ID,
    c.Genmodel,
    t.Trim,
    TRY_CAST(t.Year AS INT),
    TRY_CAST(t.Price AS NUMBER(12,2)),
    TRY_CAST(t.Gas_emission AS NUMBER(10,2)),
    t.Fuel_type,
    TRY_CAST(t.Engine_size AS NUMBER(10,2))
  FROM CARSALESFINAL.raw.TRIM t
  LEFT JOIN CARSALESFINAL.raw.CARMODEL c
    ON t.Genmodel_ID = c.Genmodel_ID
  WHERE t.Genmodel_ID IS NOT NULL;

  RETURN 'Transform CARDATA completed successfully.';
END;
$$;
```

```
-- 3. Transform CARSALES (Wide → Long format)
CREATE OR REPLACE PROCEDURE sp_transform_carsales()
RETURNS STRING
LANGUAGE SQL
AS
$$
BEGIN
  INSERT OVERWRITE INTO CARSALESFINAL.harmonized.carsales
  SELECT
    Genmodel_ID,
    Genmodel,
    Maker,
    TRY_CAST(year AS INT) AS Year,
    TRY_CAST(units_sold AS INT) AS Units_sold
  FROM CARSALESFINAL.raw.SALES
  UNPIVOT(units_sold FOR year IN (
      "2001","2002","2003","2004","2005","2006",
      "2007","2008","2009","2010","2011","2012",
      "2013","2014","2015","2016","2017","2018",
      "2019","2020"
  ));

  RETURN 'Transform CARSALES completed successfully.';
END;
$$;
```

While transform carsales converted the sales data with multiple year columns into a simpler table with one year per row, to later analyze trends more easily.

DATA TRANSFORMATION PROCEDURES

```sql
-- 5. Transform CARIMAGE
CREATE OR REPLACE PROCEDURE sp_transform_carimage()
RETURNS STRING
LANGUAGE SQL
AS
$$
BEGIN
  INSERT OVERWRITE INTO CARSALESFINAL.harmonized.carimage
  SELECT
    i.Genmodel_ID,
    c.Genmodel,
    i.Image_ID,
    i.Image_name,
    i.Predicted_viewpoint,
    i.Quality_check
  FROM CARSALESFINAL.raw.IMAGE i
  LEFT JOIN CARSALESFINAL.raw.CARMODEL c
    ON i.Genmodel_ID = c.Genmodel_ID
  WHERE i.Image_ID IS NOT NULL;

  RETURN 'Transform CARIMAGE completed successfully.';
END;
$$;
```

```sql
-- 4. Transform CARADVERTISEMENT
CREATE OR REPLACE PROCEDURE sp_transform_caradvertisement()
RETURNS STRING
LANGUAGE SQL
AS
$$
BEGIN
  INSERT OVERWRITE INTO CARSALESFINAL.harmonized.caradvertisement
  SELECT
    Adv_ID,
    Genmodel_ID,
    Genmodel,
    Maker,
    TRY_CAST(Adv_year AS INT),
    TRY_CAST(Adv_month AS INT),
    TRY_CAST(Reg_year AS INT),
    Bodytype,
    Color,
    TRY_CAST(Runned_Miles AS INT),
    TRY_CAST(Engin_size AS NUMBER(10,2)),
    Gearbox,
    Fuel_type,
    TRY_CAST(Price AS NUMBER(12,2)),
    TRY_CAST(Engine_power AS NUMBER(10,2)),
    TRY_CAST(Annual_Tax AS NUMBER(10,2)),
    TRY_CAST(Wheelbase AS NUMBER(10,2)),
    TRY_CAST(Height AS NUMBER(10,2)),
    TRY_CAST(Width AS NUMBER(10,2)),
    TRY_CAST(Length AS NUMBER(10,2)),
    TRY_CAST(Average_mpg AS NUMBER(10,2)),
    TRY_CAST(Top_speed AS NUMBER(10,2)),
    TRY_CAST(Seat_num AS SMALLINT),
    TRY_CAST(Door_num AS SMALLINT)
  FROM CARSALESFINAL.raw.ADVERTISEMENT
  WHERE Adv_ID IS NOT NULL;

  RETURN 'Transform CARADVERTISEMENT completed successfully.';
END;
$$;
```

```sql
-- 2. Transform CARPRICE (CarModel + Price)
CREATE OR REPLACE PROCEDURE sp_transform_carprice()
RETURNS STRING
LANGUAGE SQL
AS
$$
BEGIN
  INSERT OVERWRITE INTO CARSALESFINAL.harmonized.carprice
  SELECT
    p.Genmodel_ID,
    p.Maker,
    p.Genmodel,
    TRY_CAST(p.Year AS INT),
    TRY_CAST(p.Entry_price AS NUMBER(12,2))
  FROM CARSALESFINAL.raw.PRICE p
  WHERE p.Genmodel_ID IS NOT NULL;

  RETURN 'Transform CARPRICE completed successfully.';
END;
$$;
```

AUTOMATITATION OF PROCEDURES

We created a general stored procedure that called the individual transformations, to the automated it with a task that runs after the data-loading step, making the execution automatic through task orchestration.

```
--GENERAL STORE PROCEDURE 4 ALL
CREATE OR REPLACE PROCEDURE sp_transform_all()
RETURNS STRING
LANGUAGE SQL
AS
$$
BEGIN
    CALL sp_transform_cardata();
    CALL sp_transform_carprice();
    CALL sp_transform_carsales();
    CALL sp_transform_caradvertisement();
    CALL sp_transform_carimage();
    RETURN 'All transformations completed successfully.';
END;
$$;
```

```
--TASK TO USE THE STORE PROCEDURE
USE SCHEMA automation;

CREATE OR REPLACE TASK task_transform_all
    WAREHOUSE = CARSALESFINAL_WH
    AFTER task_load_sales
AS
    CALL sp_transform_all();
```

DELIVERY

CREATION OF FACT TABLES

Harmonized data is structured into facts and dimensions to enable analysis, KPIs, and dashboards.

```sql
CREATE OR REPLACE TABLE analytics.fact_sales AS
SELECT
    genmodel_id,
    genmodel,
    maker,
    year,
    units_sold
FROM harmonized.carsales;
```

Units sold per model, maker, and year.

```sql
CREATE OR REPLACE TABLE analytics.fact_price AS
SELECT
    genmodel_id,
    genmodel,
    automaker,
    year,
    entry_price
FROM harmonized.carprice;
```

Entry-level prices per model and year.

```sql
CREATE OR REPLACE TABLE analytics.fact_advertisement AS  --one row per listing
SELECT
    adv_id,
    genmodel_id,
    genmodel,
    maker,
    adv_year,
    adv_month,
    reg_year,
    bodytype,
    color,
    runned_miles,
    engin_size,
    gearbox,
    fuel_type,
    price,
    engine_power,
    annual_tax,
    wheelbase,
    height,
    width,
    length,
    average_mpg,
    top_speed,
    seat_num,
    door_num
FROM harmonized.caradvertisement;
```

One row per ad/listing with attributes like price, mileage, fuel type, dimensions.

CREATION OF DIMENSION TABLES

Harmonized data is structured into facts and dimensions to enable analysis, KPIs, and dashboards.

```sql
CREATE OR REPLACE TABLE analytics.dim_car AS
SELECT DISTINCT
    genmodel_id,
    automaker,
    automaker_id,
    genmodel,
    trim,
    fuel_type,
    engine_size,
    gas_emission
FROM harmonized.cardata;
```

Car metadata (automaker, model, trim, engine, fuel type, emissions).

```sql
CREATE OR REPLACE TABLE analytics.dim_image AS
SELECT DISTINCT
    image_id,
    genmodel_id,
    genmodel,
    image_name,
    predicted_viewpoint,
    quality_check
FROM harmonized.carimage;
```

Image metadata (id, model, viewpoint, quality).

```sql
CREATE OR REPLACE TABLE analytics.dim_date AS
SELECT DISTINCT
    year,
    month,
    CASE
      WHEN month BETWEEN 1 AND 3 THEN 1
      WHEN month BETWEEN 4 AND 6 THEN 2
      WHEN month BETWEEN 7 AND 9 THEN 3
      WHEN month BETWEEN 10 AND 12 THEN 4
    END AS quarter
FROM (
  SELECT adv_year AS year, adv_month AS month FROM harmonized.caradvertisement
  UNION
  SELECT year, NULL AS month FROM harmonized.carsales
  UNION
  SELECT year, NULL FROM harmonized.carprice
) AS tmp;
```

Calendar of years, months, and quarters derived from ads, sales, and prices.

LOAD OF FACT TABLES

```
--PROCEDURES
USE SCHEMA automation;

-- Fact Sales
CREATE OR REPLACE PROCEDURE sp_build_fact_sales()
RETURNS STRING
LANGUAGE SQL
AS
$$
BEGIN
  INSERT OVERWRITE INTO analytics.fact_sales
  SELECT * FROM harmonized.carsales;
  RETURN 'fact_sales refreshed.';
END;
$$;
```

```
-- Fact Advertisement
CREATE OR REPLACE PROCEDURE sp_build_fact_advertisement()
RETURNS STRING
LANGUAGE SQL
AS
$$
BEGIN
  INSERT OVERWRITE INTO analytics.fact_advertisement
  SELECT * FROM harmonized.caradvertisement;
  RETURN 'fact_advertisement refreshed.';
END;
$$;
```

These procedures move the transformed data from the harmonized schema into fact tables, where it's stored in a consistent way for reporting and analytics.

- Fact Sales → car models, makers, years, and units sold.
- Fact Advertisement → organized data about car advertisements.
- Fact Price → pricing details by model, year, and trim.

```
-- Fact Price
CREATE OR REPLACE PROCEDURE sp_build_fact_price()
RETURNS STRING
LANGUAGE SQL
AS
$$
BEGIN
  INSERT OVERWRITE INTO analytics.fact_price
  SELECT * FROM harmonized.carprice;
  RETURN 'fact_price refreshed.';
END;
$$;
```

LOAD OF DIMENSION TABLES

```
-- Dimension Date
CREATE OR REPLACE PROCEDURE sp_build_dim_date()
RETURNS STRING
LANGUAGE SQL
AS
$$
BEGIN
  INSERT OVERWRITE INTO analytics.dim_date
  SELECT DISTINCT
      year,
      month,
      CASE
        WHEN month BETWEEN 1 AND 3 THEN 1
        WHEN month BETWEEN 4 AND 6 THEN 2
        WHEN month BETWEEN 7 AND 9 THEN 3
        WHEN month BETWEEN 10 AND 12 THEN 4
      END AS quarter
  FROM (
    SELECT adv_year AS year, adv_month AS month FROM harmonized.caradvertisement
    UNION
    SELECT year, NULL AS month FROM harmonized.carsales
    UNION
    SELECT year, NULL FROM harmonized.carprice
  );
  RETURN 'dim_date refreshed.';
END;
$$;
```

```
-- Dimension Image
CREATE OR REPLACE PROCEDURE sp_build_dim_image()
RETURNS STRING
LANGUAGE SQL
AS
$$
BEGIN
  INSERT OVERWRITE INTO analytics.dim_image
  SELECT DISTINCT * FROM harmonized.carimage;
  RETURN 'dim_image refreshed.';
END;
$$;
```

```
-- Dimension Car
CREATE OR REPLACE PROCEDURE sp_build_dim_car()
RETURNS STRING
LANGUAGE SQL
AS
$$
BEGIN
  INSERT OVERWRITE INTO analytics.dim_car
  SELECT DISTINCT * FROM harmonized.cardata;
  RETURN 'dim_car refreshed.';
END;
$$;
```

These procedures transfer curated data from the harmonized schema into dimension tables, where it is organized to provide descriptive context for analysis.

- Dim Car → detailed information about car models, trims, and attributes.
- Dim Date → structured calendar data including years, months, and quarters for time-based analysis.
- Dim Image → references to car images that enrich the dataset with visual attributes.

BUILDING OF STAR SCHEMA

We created a master procedure that utilized both fact and dim tables, essentially forming the star schema for our analytics layer.

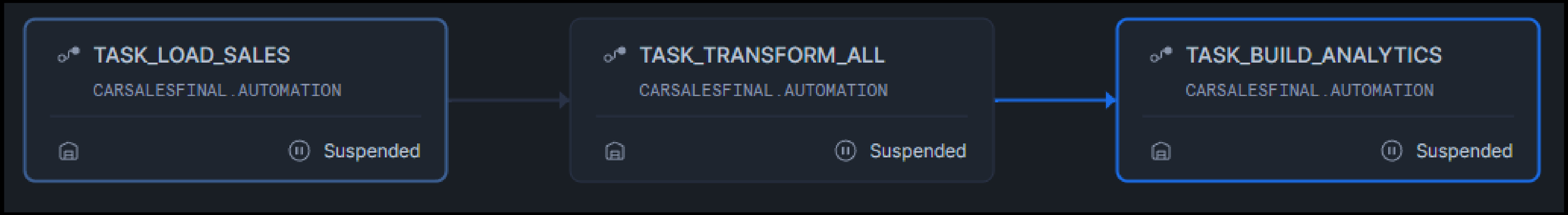
```
-- Master procedure
CREATE OR REPLACE PROCEDURE sp_build_analytics_all()
RETURNS STRING
LANGUAGE SQL
AS
$$
BEGIN
  CALL sp_build_fact_sales();
  CALL sp_build_fact_advertisement();
  CALL sp_build_fact_price();
  CALL sp_build_dim_car();
  CALL sp_build_dim_date();
  CALL sp_build_dim_image();
  RETURN 'Analytics layer fully refreshed.';
END;
$$;
```

```
--TASK
USE SCHEMA automation;

CREATE OR REPLACE TASK task_build_analytics
  WAREHOUSE = CARSALESFINAL_WH
  AFTER task_transform_all
AS
  CALL sp_build_analytics_all();
```

This process was then automated with a task, so the entire analytics layer is refreshed automatically after the transformations.

TASK ORCHESTRATION

Our pipeline runs in three linked tasks: **TASK_LOAD_SALES** detects new Sales files in S3 and lands them in raw via sp_load_sales, then it triggers **TASK_TRANSFORM_ALL** to standardize types, unpivot Sales, and build the harmonized tables, and finally **TASK_BUILD_ANALYTICS** refreshes facts, dimensions, and business views so the Streamlit dashboard always reflects the latest data without manual steps.



VIEWS

```sql
-- Top 10 cars by units sold
CREATE OR REPLACE VIEW analytics.v_top10_cars_by_sales AS
SELECT
    f.genmodel_id,
    d.genmodel,
    d.automaker,
    SUM(f.units_sold) AS total_units_sold
FROM analytics.fact_sales f
JOIN analytics.dim_car d ON f.genmodel_id = d.genmodel_id
GROUP BY f.genmodel_id, d.genmodel, d.automaker
ORDER BY total_units_sold DESC
LIMIT 10;
```

Ranks the ten models with the most units sold across the full period, showing where demand concentrates. We use it to spotlight hero models and allocate inventory and marketing accordingly.

```sql
-- Compare entry price vs advertisement price
CREATE OR REPLACE VIEW analytics.v_entry_vs_adv_price AS
SELECT
    a.maker,
    p.genmodel_id,
    d.genmodel,
    p.year,
    AVG(p.entry_price) AS avg_entry_price,
    AVG(a.price) AS avg_adv_price,
    ABS(AVG(a.price) - AVG(p.entry_price)) AS price_gap
FROM analytics.fact_price p
JOIN analytics.fact_advertisement a
    ON p.genmodel_id = a.genmodel_id
   AND p.year = a.adv_year
JOIN analytics.dim_car d
    ON p.genmodel_id = d.genmodel_id
GROUP BY a.maker, p.genmodel_id, d.genmodel, p.year
ORDER BY p.year, d.genmodel;
```

Compares official entry price with average ad price by model and year, plus the absolute gap. Positive gaps suggest scarcity or strong brand equity, negative gaps suggest discounting pressure.

```sql
-- Average advertisement price per maker
CREATE OR REPLACE VIEW analytics.v_avg_price_maker AS
SELECT
    maker,
    AVG(price) AS avg_price
FROM analytics.fact_advertisement
GROUP BY maker
ORDER BY avg_price DESC;
```

Calculates the average marketplace ad price per automaker. It reveals brand positioning in the secondary market and helps compare perceived value across makers.

```sql
--Year-overYear Sales Growth per maker
CREATE OR REPLACE VIEW analytics.v_sales_growth_maker AS
SELECT
    d.automaker,
    f.year,
    SUM(f.units_sold) AS total_units_sold,
    LAG(SUM(f.units_sold)) OVER (PARTITION BY d.automaker ORDER BY f.year) AS prev_year_units,
    ROUND(
      (SUM(f.units_sold) - LAG(SUM(f.units_sold)) OVER (PARTITION BY d.automaker ORDER BY f.year))
      / NULLIF(LAG(SUM(f.units_sold)) OVER (PARTITION BY d.automaker ORDER BY f.year), 0) * 100,
      2
    ) AS yoy_growth_pct
FROM analytics.fact_sales f
JOIN analytics.dim_car d ON f.genmodel_id = d.genmodel_id
GROUP BY d.automaker, f.year
ORDER BY d.automaker, f.year;
```

Aggregates sales by automaker and computes year over year growth with a lag window. It highlights inflection years, sustained momentum, and periods that need corrective actions.

VIEWS

```sql
--Top 10 most advertised models
CREATE OR REPLACE VIEW analytics.v_top10_most_advertised AS
SELECT
    d.automaker,
    d.genmodel,
    COUNT(*) AS num_ads
FROM analytics.fact_advertisement a
JOIN analytics.dim_car d ON a.genmodel_id = d.genmodel_id
GROUP BY d.automaker, d.genmodel
ORDER BY num_ads DESC
LIMIT 10;
```

Counts ads per model and lists the top ten by volume, grouped with their automaker. Use it to see which models get the most marketplace visibility and where sellers, dealers, or owners are concentrating attention.

```sql
--Top 10 most advertised models vs top 10 most sold models

CREATE OR REPLACE VIEW analytics.v_top10_most_advs_vs_most_sold AS
WITH top_ads AS (
    SELECT
        a.genmodel_id,
        COUNT(*) AS total_ads
    FROM analytics.fact_advertisement a
    GROUP BY a.genmodel_id
    ORDER BY total_ads DESC
    LIMIT 10
),
top_sales AS (
    SELECT
        f.genmodel_id,
        SUM(f.units_sold) AS total_units_sold
    FROM analytics.fact_sales f
    GROUP BY f.genmodel_id
    ORDER BY total_units_sold DESC
    LIMIT 10
)
SELECT
    d.automaker,
    d.genmodel,
    COALESCE(sa.total_units_sold, 0) AS total_units_sold,
    COALESCE(ad.total_ads, 0) AS total_ads
FROM analytics.dim_car d
LEFT JOIN top_sales sa ON d.genmodel_id = sa.genmodel_id
LEFT JOIN top_ads ad ON d.genmodel_id = ad.genmodel_id
WHERE sa.genmodel_id IS NOT NULL OR ad.genmodel_id IS NOT NULL
ORDER BY total_ads DESC, total_units_sold DESC;
```

Builds two top-10 lists, one by total ads and one by total units sold, then joins them to compare exposure versus actual demand. It highlights models where advertising converts into sales and models that are over promoted relative to results.

RESULT

STREAMLIT

```python
import streamlit as st
import altair as alt
from snowflake.snowpark.context import get_active_session
import pandas as pd

session = get_active_session()

st.title("Car Sales - Year-over-Year Growth Dashboard")
```

1. Import required libraries, connect to the Snowflake session, and set the app title.

```python
def load_view(view_name):
    df = session.table(view_name).to_pandas()
    df.columns = [col.lower() for col in df.columns]
    return df

sales_growth_df = load_view("analytics.v_sales_growth_maker")
```

2. Define a function to load the Snowflake view and convert it into a pandas DataFrame.

```python
automakers = st.multiselect(
    "Select makers (Automakers):",
    options=sales_growth_df["automaker"].unique(),
    default=[]
)

filtered_df = sales_growth_df[sales_growth_df["automaker"].isin(automakers)]
```

3. Lets the user choose which automakers to analyze and filters the dataset accordingly.

STREAMLIT

```python
st.header("YoY Growth evolution per maker")

if not filtered_df.empty:
    max_growth = int(filtered_df["yoy_growth_pct"].max())
    growth_limit = st.slider(
        "Filter makers with max YoY growth (%)",
        min_value=-100,
        max_value=max_growth,
        value=max_growth,
        step=10
    )

    max_year = int(filtered_df["year"].max())
    min_year = int(filtered_df["year"].min())

    year_range = st.slider(
        "Select year range",
        min_value=min_year,
        max_value=max_year,
        value=(min_year, max_year),
        step=1
    )
```

4. Adds sliders to filter by maximum YoY growth percentage and select a year range for analysis.

```python
    filtered_chart_df = filtered_df[
        (filtered_df["year"] >= year_range[0]) &
        (filtered_df["year"] <= year_range[1]) &
        (filtered_df["yoy_growth_pct"] <= growth_limit)
    ].sort_values(["automaker", "year"])

    chart = alt.Chart(filtered_chart_df).mark_line(point=True).encode(
        x=alt.X("year:O", title="Año"),
        y=alt.Y("yoy_growth_pct:Q", title="Growth YoY (%)"),
        color="automaker:N",
        tooltip=["automaker", "year", "total_units_sold", "prev_year_units", "yoy_growth_pct"]
    ).properties(
        width=800,
        height=500
    )

    st.altair_chart(chart, use_container_width=True)
else:
    st.text("Select a maker to start")
```

5. Apply the filters to the dataset, then create and display a line chart showing YoY growth by automaker.

STREAMLIT

Car Sales - Year-over-Year Growth Dashboard

Select makers (Automakers):

BMW × Mazda × MG ×

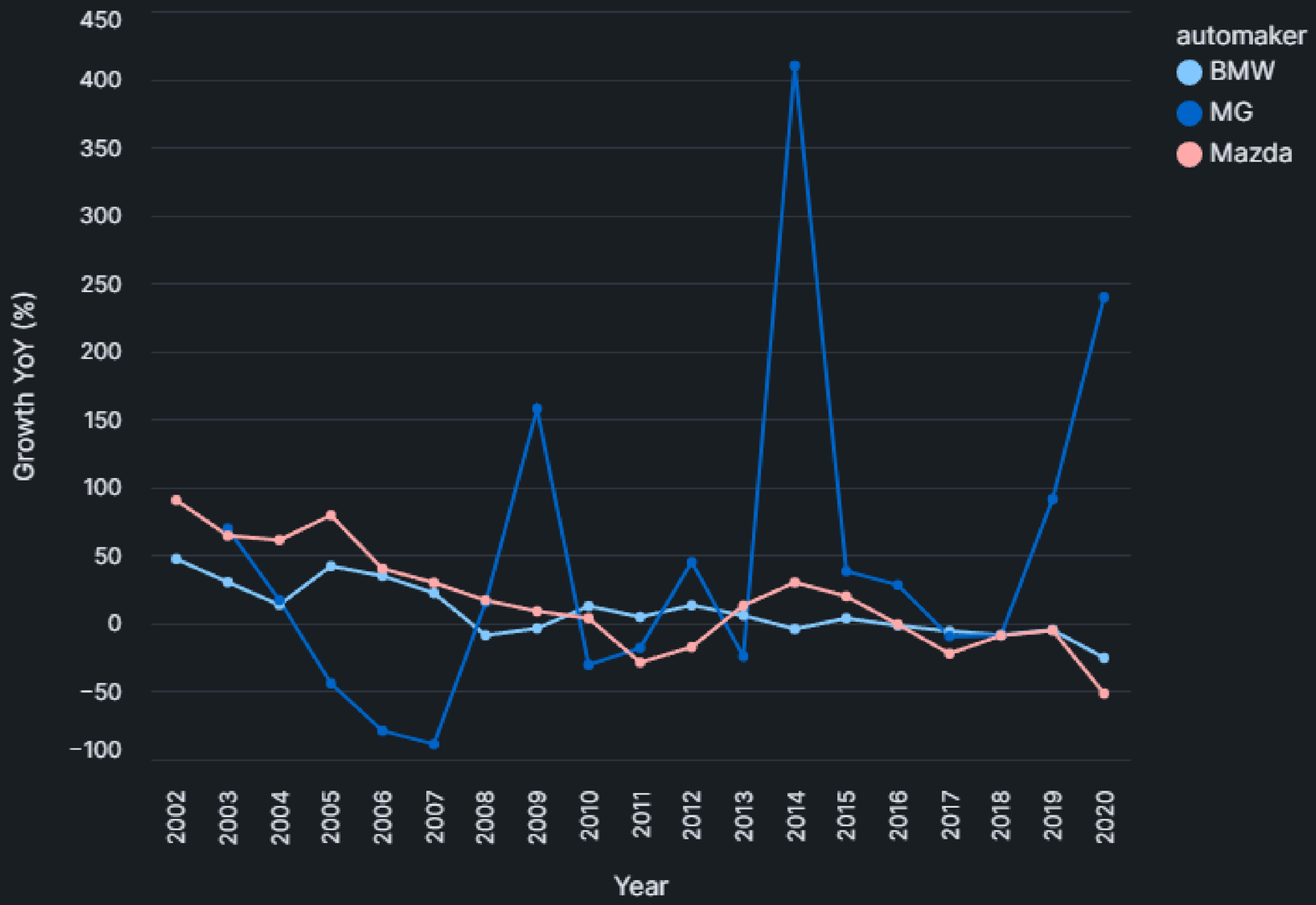
YoY Growth evolution per maker

Filter makers with max YoY growth (%)

644

Select year range

2001 2020



This Streamlit dashboard lets users filter automakers and years to visualize their year-over-year sales growth in an interactive line chart.

New Technologies in Informatics

THANK YOU