

PROYECTO INTEGRADOR III

Diego Lopez Castan



INDICE

- Objetivo Proyecto
- Diagrama de Arquitectura
- Justificación de Tecnologias
- Preguntas de negocio
- KPIs
- Fuentes de Datos
- Arquitectura
- Modelado Dimensional
- Transformaciones y DBT

OBJETIVO



Este proyecto surge para dotar a una empresa en fuerte crecimiento de una plataforma de datos confiable, escalable y gobernada. Nuestro objetivo es unificar, organizar y transformar grandes volúmenes de información provenientes de múltiples fuentes, a fin de acelerar la toma de decisiones estratégicas con métricas consistentes y trazables.

Diseñaremos e implementaremos un pipeline ELT y un Data Warehouse en capas (raw, staging, core, gold) que consolide la información, reduzca la dependencia del equipo técnico y empodere a analistas y áreas de negocio para crear reportes, KPIs e insights con autonomía.

OBJETIVO DEL PIPELINE



Implementar un pipeline ELT robusto, escalable y auditabile que permita:

- Ingerir datos de fuentes internas y externas (bases OLTP, archivos CSV/Parquet, APIs y web scraping).
- Persistir los datos crudos en una landing zone y en la capa raw del DWH.
- Estandarizar y transformar en capas (staging → core → marts) usando dbt, con pruebas de calidad.
- Exponer datasets listos para análisis (BI/ML) y gobernados (catálogo, linaje, control de accesos).
-

KPIs esperados (ejemplos): pricing competitivo por zona y tipo de habitación, oferta vs. disponibilidad, intensidad de demanda (reviews/mes), concentración por host, estacionalidad y mix de producto.

DIAGRAMA DE ARQUITECTURA

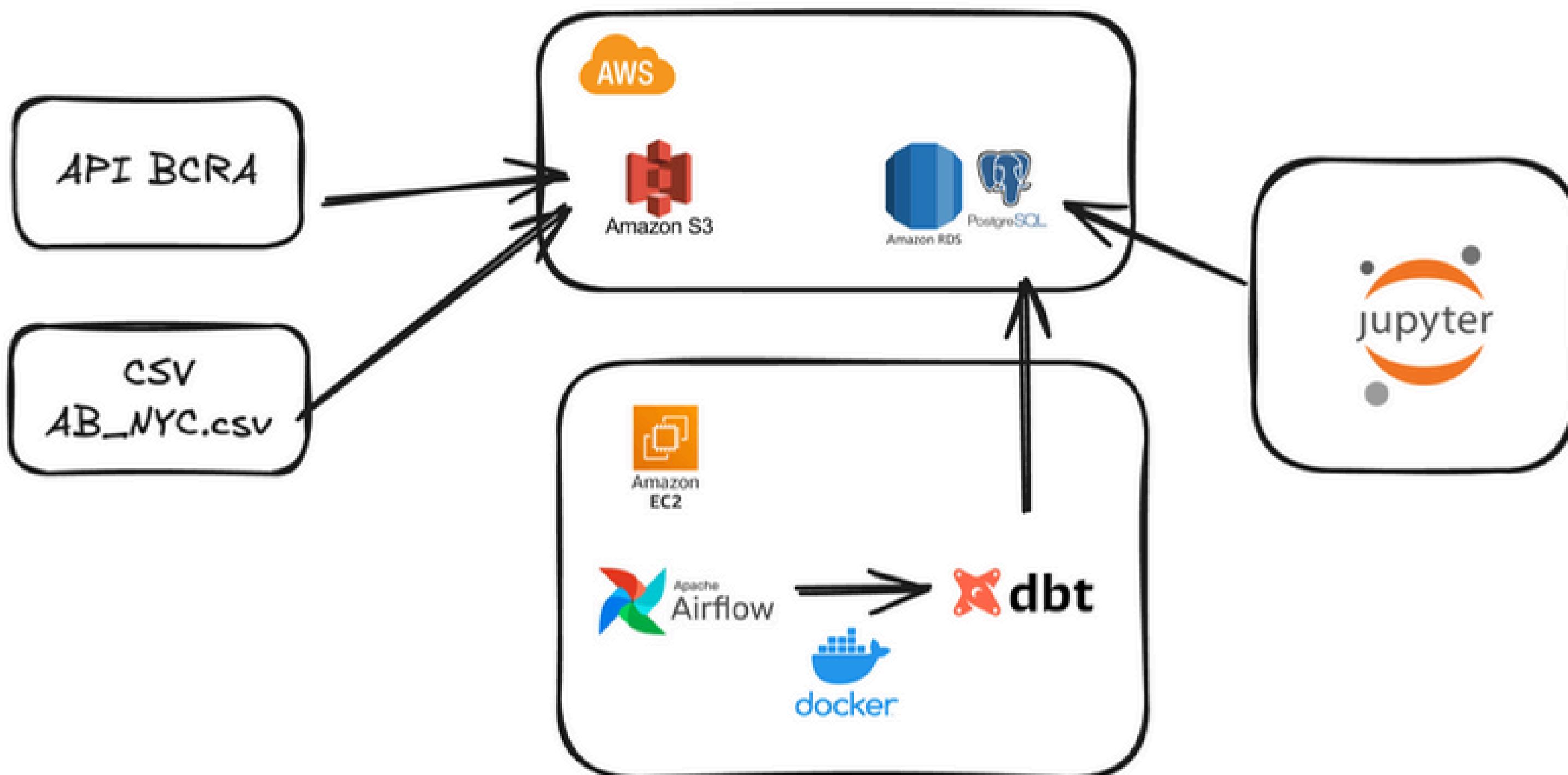
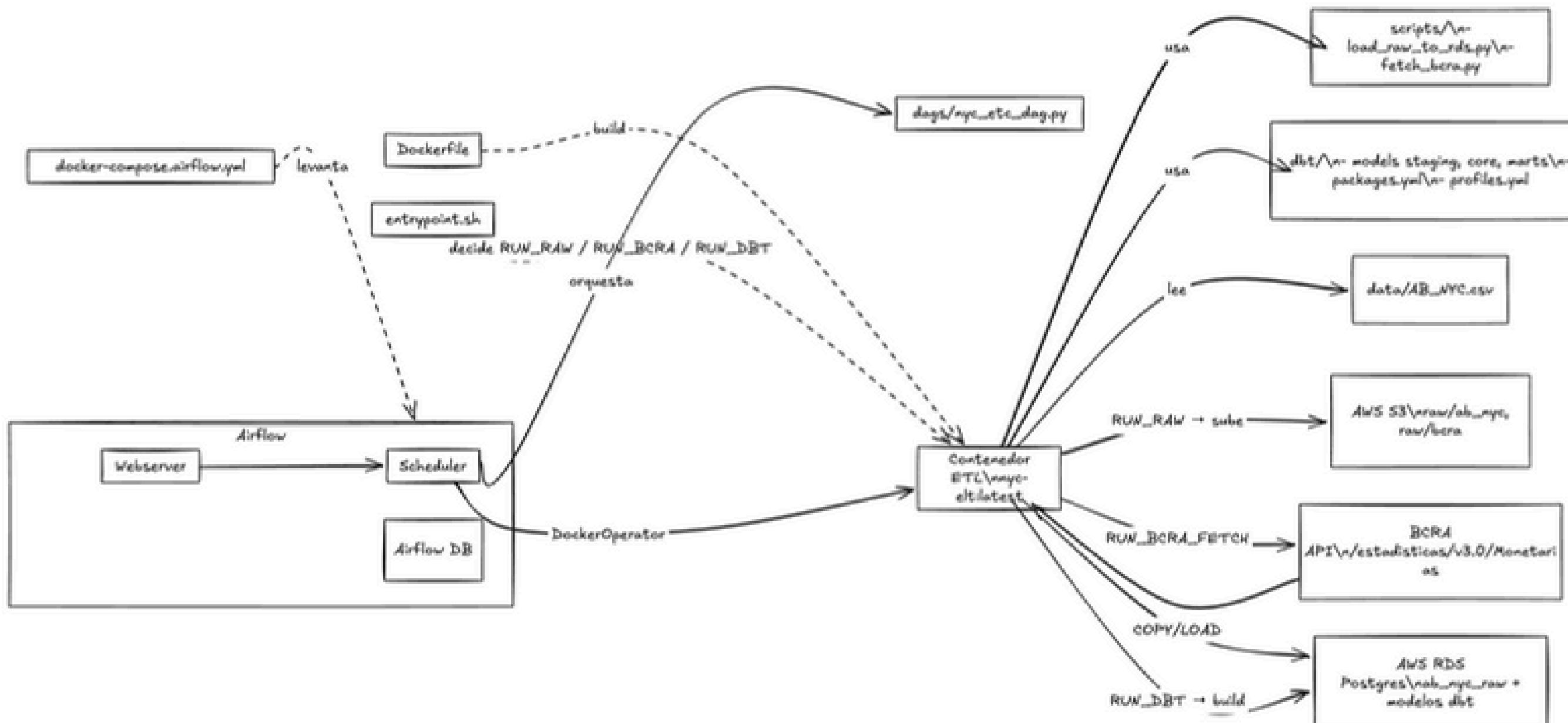


DIAGRAMA TÉCNICO DE ARQUITECTURA



JUSTIFICACIÓN DE TECNOLOGIAS



Python: para extracción, transformación liviana/mediana y glue code entre servicios.

Por qué?: Ecosistema maduro para datos (pandas, requests, sqlalchemy, boto3) y comunidad enorme.

Beneficios clave:

- Velocidad de desarrollo: mucho resuelto con librerías probadas.
- Conectores listos: HTTP/REST, bases SQL/NoSQL, S3, etc.

Apache Airflow: para la Orquestación de ETL/ELT, dependencias, reintentos, alertas y programación.

Por qué?: es un estándar, tiene UI clara y extensible por operators/hooks.

Beneficios clave:

- Reproducibilidad temporal: DAGs versionados, backfills controlados.
- Confiabilidad: reintentos, SLA, sensores; fácil integrar alertas (Slack/Email).
- Observabilidad: Gantt, Tree View, logs por tarea.

JUSTIFICACIÓN DE TECNOLOGIAS



Docker: para crear contenedores con portabilidad y entornos idénticos en dev/QA/prod.

Por qué: Empaque Airflow, dbt y jobs de Python con sus dependencias.

Beneficios clave:

- Deploy simple: imágenes versionadas; rollbacks rápidos.
- Aislamiento: menos conflictos entre librerías/sistemas.

AWS S3 (Data Lake RAW): para almacenamiento económico, de datos crudos.

Por qué: Costos bajos vs. discos/DB, integración nativa con AWS y herramientas de datos.

Beneficios clave:

- Versionado de archivos: historial completo y recuperación ante errores.
- Particionado por fecha: performance en lecturas selectivas y orden operacional.

JUSTIFICACIÓN DE TECNOLOGIAS



AWS RDS con Postgres: para las tablas y vistas Staging/Core/Marts

Por qué?: SQL estándar, ecosistema robusto (extensiones), “managed” con backups/patching.

Beneficios clave:

- Consistencia & ACID: base sólida para joins y reglas de negocio.
- Escalabilidad vertical-horizontal (read replicas): soporta crecimiento de lecturas.
- Mantenibilidad: menos overhead operativo que autogestionar Postgres.

dbt: para transformaciones, documentacion y tests.

Por qué?: Estándar moderno de ELT; convierte SQL en un proyecto de software.

Beneficios clave:

- Versionado y revisiones: modelos como código (git).
- Calidad: tests (unique, not_null, relationships) y macros reutilizables.
- Documentación: dbt docs con linaje y descripciones.

Jupyter Notebook: para exploración y análisis de los datos.

Por qué?: Feedback inmediato y narrativa (código + gráficos + texto).

Beneficios clave:

- Integración: conecta a Postgres (SQLAlchemy/psycopg) y lee S3.
- Buenas prácticas: notebooks “exploratorios” separados de “productivos”, semillas/fixtures para reproducibilidad, exportar hallazgos a scripts/dbt.

DEFINICIÓN DATA LAKE



- Propósito: almacenamiento inmutable de datos crudos, tal como llegan (formato y esquema originales).
- Características: partición por fecha de ingestión (ingestion_date), schema-on-read, no lógicas de negocio.
- Ejemplos: raw_ab_nyc_csv_yyyymmdd.parquet en S3; tabla raw.ab_nyc en Postgres.

DEFINICIÓN DATA WAREHOUSE



Staging

- Propósito: normalizar tipos y nombres, limpiar valores obvios (trimming, casting, fechas), deduplicar e introducir claves técnicas (surrogate keys).
- Convenciones: prefijo stg_ y una tabla por fuente dominante.
- Ejemplo: stg_listings (a partir de raw.ab_nyc).

Core

- Propósito: modelo canónico orientado a negocio (dimensiones y hechos), integrando múltiples fuentes.
- Ejemplo de entidades: dim_host, dim_neighbourhood, dim_room_type, fact_listing_snapshot, fact_reviews_monthly.

Gold / Marts (Consumo)

- Propósito: datasets listos para BI/ML, con métricas aprobadas y grain definido.
- Ejemplos: mart_pricing_by_zone, mart_supply_vs_availability, mart_host_concentration, mart_demand_trends.

PREGUNTAS DE NEGOCIO

Algunas preguntas que nos fueron enviadas son:



- ¿Cuál es el precio promedio de los alojamientos por barrio y distrito?
- ¿Qué tipo de habitación es el más ofrecido y cuál genera mayor revenue estimado?
- ¿Cuáles son los anfitriones con más propiedades listadas y cómo varían sus precios?
- ¿Existen diferencias significativas en la disponibilidad anual entre barrios o tipos de alojamiento?
- ¿Cómo evoluciona el número de reseñas por mes en los diferentes distritos de la ciudad?
- ¿Qué barrios tienen la mayor concentración de alojamientos activos?
- ¿Cómo se distribuyen los precios y qué outliers existen?
- ¿Qué relación hay entre la disponibilidad anual y la cantidad de reseñas como proxy de ocupación?

PREGUNTAS EXTRA

Algunas preguntas que nos fueron enviadas son:

- ¿Cuales son los alojamientos con más de reseñas?
- ¿Cuál es el ingreso promedio estimado por anfitrión en cada distrito?
- ¿Qué porcentaje de alojamientos están subvaluados o sobrevaluados respecto al promedio de su barrio?
- ¿Qué distritos tienen mayor disponibilidad anual promedio?
- ¿Qué barrios tienden a ofrecer disponibilidad completa (365 días) frente a disponibilidad parcial?
- ¿Qué proporción de propiedades tiene disponibilidad muy baja (<30 días)?



FUENTES DE DATOS



Para llevar a cabo los diferentes análisis del proyecto, se nos proporcionó un archivo CSV con información de alojamientos en la ciudad de Nueva York. Dicho archivo contiene los siguientes campos:

id: Identificador único del alojamiento.

name: Nombre del anuncio.

host_id: Identificador único del anfitrión.

host_name: Nombre del anfitrión.

neighbourhood_group: Distrito grande de Nueva York

neighbourhood: Barrio específico dentro del distrito.

latitude: Latitud de la propiedad.

longitude: Longitud de la propiedad.

room_type: Tipo de habitación

price: Precio por noche en dólares.

minimum_nights: Número mínimo de noches requeridas para reservar.

number_of_reviews: Total de reseñas recibidas.

last_review: Fecha de la última reseña.

reviews_per_month: Reseñas promedio por mes.

calculated_host_listings_count: Número de propiedades que el anfitrión tiene listadas.

availability_365: Días disponibles para reservar en un año.

TABLAS DIMENSIONES



Tabla: dim_date

Descripción: Dimensión de calendario que permite analizar datos a lo largo del tiempo en diferentes jerarquías (año, trimestre, mes, semana, día).

date_sk: Clave surrogate de la fecha en formato yyyyymmdd. PK de la tabla.

full_date: Fecha completa en tipo DATE. Valor único.

year: Año correspondiente a la fecha (YYYY).

quarter: Trimestre del año (1 a 4).

month: Mes del año (1 a 12).

day: Día del mes (1 a 31).

day_of_week: Día de la semana en formato ISO (1=Lunes, 7=Domingo).

week_of_year: Semana ISO del año (1 a 52/53).

is_weekend: Indicador booleano, TRUE si la fecha corresponde a sábado o domingo.

TABLAS DIMENSIONES

Tabla: dim_host

Descripción: Dimensión de anfitriones. Contiene atributos del host y métricas agregadas (como cantidad de listings).

host_sk: Clave surrogate generada automáticamente (BIGSERIAL). PK de la tabla.

host_id: Identificador natural del host (fuente de datos). Debe ser único.

host_name: Nombre del anfitrión. Puede contener datos personales (PII), considerar anonimizar.

listings_count: Número de listings que tiene el host en la fuente.

is_multilister: Indicador booleano, TRUE si el host tiene ≥ 5 listings (considerado multianfitrión).

updated_at: Fecha y hora de última actualización del registro.



TABLAS DIMENSIONES



Tabla: dim_location

Descripción: Dimensión de ubicación geográfica. Define el barrio y borough de cada listing.

location_sk: Clave surrogate generada automáticamente. PK de la tabla.

neighbourhood_group: Borough donde se ubica el listing (ej. Manhattan, Brooklyn).

neighbourhood: Barrio específico dentro del borough.

Tabla: dim_room_type

Descripción: Catálogo de tipos de habitación ofertados en los listings.

room_type_sk: Clave surrogate generada automáticamente. PK de la tabla.

room_type: Tipo de habitación (ejemplo: "Entire home/apt", "Private room", "Shared room").

Valor único.

TABLAS HECHOS

Tabla: fact_listing_snapshot

Descripción: Tabla de hechos principal. Registra el estado de cada listing por día (snapshot), incluyendo precios, disponibilidad y reseñas.

listing_id: Identificador natural del listing (del CSV). Parte de la PK junto con date_sk.

date_sk: Clave surrogate de fecha. FK hacia dim_date.

host_sk: Clave surrogate de host. FK hacia dim_host.

location_sk: Clave surrogate de ubicación. FK hacia dim_location.

room_type_sk: Clave surrogate de tipo de habitación. FK hacia dim_room_type.

price: Precio por noche en USD. No puede ser negativo.

minimum_nights: Número mínimo de noches requeridas en la reserva. No puede ser menor a 1.

number_of_reviews: Total acumulado de reseñas para el listing.

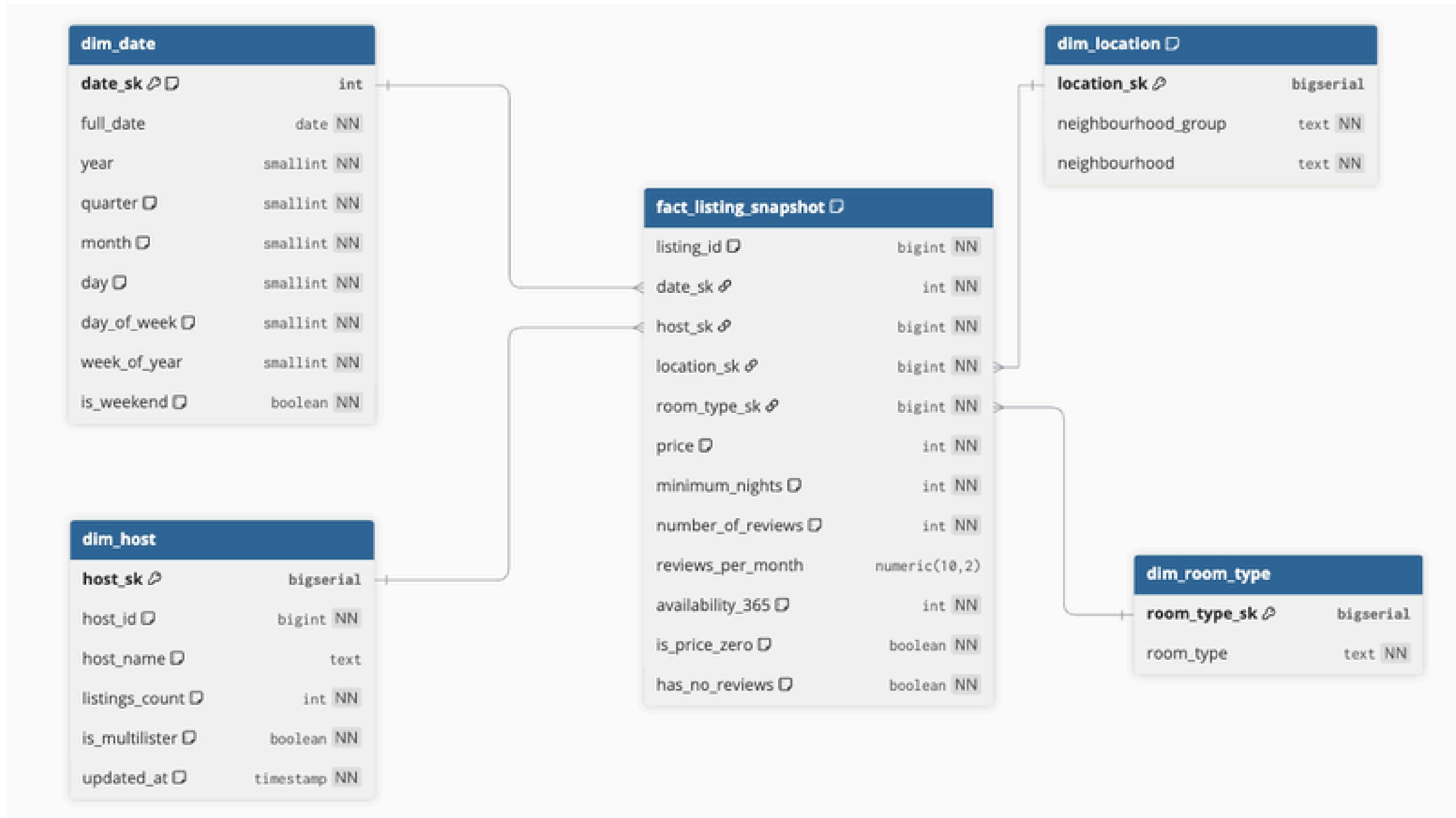
reviews_per_month: Promedio de reseñas recibidas por mes. Puede ser nulo si no existen reseñas.

availability_365: Número de días disponibles en el año (rango 0 a 365).

is_price_zero: Flag booleano. TRUE si el precio es 0 (posible anomalía).

has_no_reviews: Flag booleano. TRUE si el listing no tiene reseñas (reviews_per_month nulo).





TRANSFORMACIONES Y DBT

La arquitectura medallón es un paradigma de lakehouse que organiza los datos en tres capas progresivas (Bronze, Silver y Gold), donde cada nivel agrega valor y refinamiento a los datos. Esta aproximación permite un balance óptimo entre flexibilidad, calidad y performance, facilitando tanto el análisis exploratorio como los casos de uso empresariales críticos.

Beneficios de la Arquitectura Medallón

Escalabilidad y Flexibilidad:

- Separación clara de responsabilidades permite equipos especializados
- Cada capa puede escalar independientemente según la demanda
- Facilita la incorporación de nuevas fuentes de datos

Calidad y Gobernanza:

- Control progresivo de calidad en cada capa
- Trazabilidad completa del linaje de datos
- Implementación consistente de políticas de datos

Performance y Eficiencia:

- Optimización específica por caso de uso
- Reutilización de transformaciones entre equipos
- Reducción de carga computacional en consultas finales

BRONZE LAYER (STAGING)

La capa Staging actúa como el primer nivel de transformación sobre los datos crudos. Toma las tablas provenientes de la capa Bronze (en este caso la fuente raw.ab_nyc) y aplica procesos de limpieza, estandarización y control de calidad para dejarlos listos para integraciones posteriores.

Características principales

- Estandarización de tipos de datos: Conversión explícita de campos como host_id, price, latitude, longitude, number_of_reviews y availability_365 a tipos numéricos apropiados.
- Normalización de valores faltantes o vacíos: Uso de nullif(trim(...),"") para reemplazar cadenas vacías por NULL en columnas como host_name, neighbourhood_group, neighbourhood y room_type.
- Reglas de consistencia en métricas clave:
 - minimum_nights: Se asegura un valor mínimo de 1.
 - availability_365: Se limita el rango a entre 0 y 365.
 - reviews_per_month: Se castea a numeric(10,2) para preservar decimales.
- Control de calidad con tests (stg_listings.yml):
 - id: no nulo y único.
 - price: valores mayores o iguales a 0.
 - availability_365: valores entre 0 y 365.
- Registro temporal: Se agrega snapshot_date = current_date para permitir trazabilidad de cuándo se tomó la foto de los datos.

Casos de uso:

- Auditoría y compliance regulatorio
- Análisis de datos
- Recuperación ante errores en capas superiores
- Exploración de datos no estructurados

CORE LAYER (INTERMEDIATE)

La capa Core constituye el modelo dimensional del proyecto. A partir de los datos limpios y estandarizados de Staging, aquí se construyen dimensiones y tablas de hechos que permiten consultas eficientes y consistentes para análisis de negocio.

Características principales

- Modelo en estrella (Star Schema):
- Organiza los datos en dimensiones (descriptivas) y hechos (eventos medibles).
- Integridad referencial garantizada:
- Todas las claves foráneas de la tabla de hechos (`fact_listing_snapshot`) tienen relaciones validadas con las claves primarias de sus respectivas dimensiones.
- Optimización para análisis:
- Las dimensiones incluyen atributos enriquecidos (ej. `is_multilister` en `hosts`) y la fact table concentra métricas clave como `price`, `minimum_nights` y `availability_365`.
- Control de calidad de datos:
- Tests automáticos en DBT (`not_null`, `unique`, `relationships`, `expectations`) aseguran la consistencia y validez de métricas.

Casos de uso generales de la capa Core

- Base consistente para KPIs en la capa Mart.
- Análisis multidimensional por host, ubicación, tipo de habitación y tiempo.
- Soporte a dashboards de monitoreo de tendencias y métricas clave.
- Confiabilidad analítica gracias a la validación de integridad y reglas de negocio.

CORE LAYER (INTERMEDIATE)

Modelos de Dimensión:

dim_date.sql: Genera un calendario completo del 2015 al 2035. Incluye claves de fecha (date_sk), atributos de granularidad (año, mes, semana, día) y banderas como is_weekend.

dim_host.sql: Estandariza información de anfitriones (host_id) y agrega atributos:

- listings_count: cantidad de propiedades publicadas.
- is_multilister: flag para identificar si un host tiene 5+ propiedades.
- updated_at: marca de última actualización.
- Casos de uso: segmentación de anfitriones, estudios de oferta profesionalizada.

dim_location.sql: Normaliza ubicaciones con neighbourhood_group y neighbourhood. Cada combinación genera una clave única location_sk.

dim_room_type.sql: Estandariza categorías de tipo de habitación (room_type), generando una clave surrogate room_type_sk.

CORE LAYER (INTERMEDIATE)

Tabla de Hechos

fact_listing_snapshot.sql: Consolida la información de un snapshot de anuncios de Airbnb:

- Métricas principales: price, minimum_nights, number_of_reviews, reviews_per_month, availability_365.
- Flags de calidad: is_price_zero, has_no_reviews.
- Claves foráneas: date_sk, host_sk, location_sk, room_type_sk, que vinculan con las dimensiones.

Casos de uso:

- Análisis de precios y disponibilidad en el tiempo.
- Estudio de demanda a partir de reseñas.
- Identificación de patrones de oferta por ubicación y tipo de habitación.

GOLD LAYER (DATA MARTS)

La capa Marts constituye la capa de negocio (Gold) del modelo. A partir de las dimensiones y hechos de la capa Core, construye vistas analíticas que exponen indicadores clave (KPIs) y métricas directamente consumibles por analistas, dashboards o reportes.

Características principales

- Agregación orientada al negocio: métricas de precios, disponibilidad, concentración de listings y reseñas.
- Optimización para consumo: todos los modelos se materializan como views, lo que permite consultas rápidas sin duplicar almacenamiento.
- Variedad de perspectivas analíticas: métricas por ubicación, host, tipo de habitación y tendencias temporales.
- Control de documentación en DBT: cada modelo cuenta con descripciones en marts.yml que facilitan la gobernanza y el entendimiento de métricas.

Casos de uso generales de la capa Marts

- Dashboards ejecutivos: KPIs de precios, disponibilidad, reseñas y concentración.
- Toma de decisiones comerciales: estrategias de pricing, gestión de inventario y expansión geográfica.
- Detección de oportunidades: barrios emergentes, tipos de habitación con mayor revenue potencial.
- Monitoreo de tendencias: evolución temporal de demanda y oferta.

GOLD LAYER (DATA MARTS)

Modelos y métricas principales

`mart_avg_price_by_area`: Precio promedio por barrio y distrito.

Casos de uso: analizar diferencias de precios entre áreas; detectar zonas premium o low-cost.

`mart_roomtype_supply_and_rev`: Cantidad de listings por tipo de habitación y revenue estimado (proxy) calculado con ocupación x precio.

Casos de uso: análisis de rentabilidad por modalidad de hospedaje (entero, privado, compartido).

`mart_host_inventory_and_prices`: Top hosts por cantidad de listings, con estadísticas de precio (avg_price, p50_price, std_price)

Casos de uso: identificar superhosts, concentración de mercado y variabilidad de precios por host.

`mart_availability_by_area_and_type`: Disponibilidad promedio (availability_365) segmentada por distrito, barrio y tipo de habitación.

Casos de uso: medir oferta activa y patrones de disponibilidad en distintas categorías.

GOLD LAYER (DATA MARTS)

`mart_reviews_trend_by_district`

Tendencia mensual de reseñas agregadas por distrito.

Casos de uso: proxy de demanda; permite observar estacionalidad y crecimiento de reseñas.

`mart_active_concentration_by_neighbourhood`: Conteo de listings activos (con precio > 0 y disponibilidad > 0) por barrio.

Casos de uso: detectar barrios con alta concentración de oferta; estudios de saturación de mercado.

`mart_price_distribution_outliers`: Distribución de precios por distrito, con cálculo de percentiles (p25, p50, p75) y detección de outliers altos/bajos vía rango intercuartílico.

Casos de uso: análisis de dispersión de precios, detección de anomalías o estrategias de pricing extremas.

`mart_availability_vs_reviews`: Correlación entre ocupación (proxy: 365 - availability_365) y reseñas por distrito.

Casos de uso: medir si mayor ocupación está asociada a mayor volumen de reseñas (relación oferta vs. demanda).

VALIDACION DBT LOCAL



```
22:08:33  Installing metaplane/dbt_expectations
22:08:34  Installed from version 0.10.9
22:08:34  Up to date!
22:08:34  Installing godatadriven/dbt_date
22:08:35  Installed from version 0.15.0
22:08:35  Up to date!
Core:
- installed: 1.7.13
- latest: 1.10.10 - Update available!

Your version of dbt-core is out of date!
You can find instructions for upgrading here:
https://docs.getdbt.com/docs/installation

Plugins:
- postgres: 1.7.13 - Update available!

At least one plugin is out of date or incompatible with dbt-core.
You can find instructions for upgrading here:
https://docs.getdbt.com/docs/installation

22:08:38  Running with dbt=1.7.13
22:08:39  Registered adapter: postgres=1.7.13
22:08:39  Unable to do partial parsing because saved manifest not found. Starting full parse.
22:08:40  Found 14 models, 20 tests, 1 source, 0 exposures, 0 metrics, 793 macros, 0 groups, 0 semantic models
22:08:40
22:08:44  Concurrency: 4 threads (target='prod')
22:08:44
22:08:44  1 of 20 START test dbt_expectations_expect_column_values_to_be_between_fact_listing_snapshot_availability_365_365_0 [RUN]
22:08:44  2 of 20 START test dbt_expectations_expect_column_values_to_be_between_fact_listing_snapshot_minimum_nights_1 [RUN]
22:08:44  3 of 20 START test dbt_expectations_expect_column_values_to_be_between_fact_listing_snapshot_price_0 [RUN]
22:08:44  4 of 20 START test dbt_expectations_expect_column_values_to_be_between_stg_listings_availability_365_365_0 [RUN]
22:08:46  1 of 20 PASS dbt_expectations_expect_column_values_to_be_between_fact_listing_snapshot_availability_365_365_0 [PASS in 1.94s]
22:08:46  4 of 20 PASS dbt_expectations_expect_column_values_to_be_between_stg_listings_availability_365_365_0 [PASS in 1.94s]
22:08:46  3 of 20 PASS dbt_expectations_expect_column_values_to_be_between_fact_listing_snapshot_price_0 [PASS in 1.94s]
22:08:46  2 of 20 PASS dbt_expectations_expect_column_values_to_be_between_fact_listing_snapshot_minimum_nights_1 [PASS in 1.95s]
22:08:46  5 of 20 START test dbt_expectations_expect_column_values_to_be_between_stg_listings_price_0 [RUN]
22:08:46  6 of 20 START test not_null_dim_date_date_sk ..... [RUN]
22:08:46  7 of 20 START test not_null_dim_host_host_sk ..... [RUN]
22:08:46  8 of 20 START test not_null_dim_location_location_sk ..... [RUN]
22:08:48  5 of 20 PASS dbt_expectations_expect_column_values_to_be_between_stg_listings_price_0 [PASS in 1.92s]
22:08:48  8 of 20 PASS not_null_dim_location_location_sk ..... [PASS in 1.92s]
22:08:48  9 of 20 START test not_null_dim_room_type_room_type_sk ..... [RUN]
22:08:48  10 of 20 START test not_null_fact_listing_snapshot_listing_id ..... [RUN]
22:08:48  6 of 20 PASS not_null_dim_date_date_sk ..... [PASS in 1.94s]
22:08:48  11 of 20 START test not_null_stg_listings_id ..... [RUN]
22:08:48  7 of 20 PASS not_null_dim_host_host_sk ..... [PASS in 2.05s]
22:08:48  12 of 20 START test relationships_fact_listing_snapshot_date_sk_date_sk_ref_dim_date_ [RUN]
22:08:50  9 of 20 PASS not_null_dim_room_type_room_type_sk ..... [PASS in 1.86s]
22:08:50  13 of 20 START test relationships_fact_listing_snapshot_host_sk_host_sk_ref_dim_host_ [RUN]
22:08:50  10 of 20 PASS not_null_fact_listing_snapshot_listing_id ..... [PASS in 1.85s]
22:08:50  14 of 20 START test relationships_fact_listing_snapshot_location_sk_ref_dim_location_ [RUN]
22:08:50  11 of 20 PASS not_null_stg_listings_id ..... [PASS in 1.94s]
22:08:50  15 of 20 START test relationships_fact_listing_snapshot_room_type_sk_room_type_sk_ref_dim_room_type_ [RUN]
22:08:50  12 of 20 PASS relationships_fact_listing_snapshot_date_sk_date_sk_ref_dim_date_ [PASS in 1.85s]
22:08:50  16 of 20 START test unique_dim_date_date_sk ..... [RUN]
22:08:52  14 of 20 PASS relationships_fact_listing_snapshot_location_sk_ref_dim_location_ [PASS in 1.87s]
22:08:52  17 of 20 START test unique_dim_host_host_sk ..... [RUN]
22:08:52  13 of 20 PASS relationships_fact_listing_snapshot_host_sk_host_sk_ref_dim_host_ [PASS in 1.89s]
22:08:52  18 of 20 START test unique_dim_location_location_sk ..... [RUN]
22:08:52  16 of 20 PASS unique_dim_date_date_sk ..... [PASS in 1.85s]
22:08:52  15 of 20 PASS relationships_fact_listing_snapshot_room_type_sk_room_type_sk_ref_dim_room_type_ [PASS in 1.86s]
22:08:52  19 of 20 START test unique_dim_room_type_room_type_sk ..... [RUN]
22:08:52  20 of 20 START test unique_stg_listings_id ..... [RUN]
22:08:54  18 of 20 PASS unique_dim_location_location_sk ..... [PASS in 1.85s]
22:08:54  17 of 20 PASS unique_dim_host_host_sk ..... [PASS in 1.87s]
22:08:54  19 of 20 PASS unique_dim_room_type_room_type_sk ..... [PASS in 1.84s]
22:08:54  20 of 20 PASS unique_stg_listings_id ..... [PASS in 1.89s]
22:08:55
22:08:55  Finished running 20 tests in 0 hours 0 minutes and 15.58 seconds (15.58s).
22:08:55
22:08:55  Completed successfully
22:08:55
22:08:55  Done. PASS=20 WARN=0 ERROR=0 SKIP=0 TOTAL=20
```

AIRFLOW

14:55 UTC - AF

DAG: nyc_elt_bcra_dbt ELT: AD_NYC > S3/RDS, BCRA > S3, bcra dbt

Schedule: 0 9 * * * | Next Run ID: 2025-08-30, 09:00:00 UTC | [Run](#) | [Cancel](#)

31/08/2025 | 12:10:44 a.m. | All Run Types | All Run States | Clear Filters | Auto-refresh: 25 | No status

Press Shift + F for Shortcuts | [detained](#) | [failed](#) | [queued](#) | [removed](#) | [restarting](#) | [running](#) | [scheduled](#) | [shutdown](#) | [skipped](#) | [success](#) | [up_for_reschedule](#) | [up_for_retry](#) | [upstream_failed](#) | No status

Task: raw_to_s3_rds

Clear task | Mark state as... | Filter DAG by task

Details | Graph | Gantt | Code | Audit Log | Logs | XCom | Task Duration

Layout: Left to Right

raw_to_s3_rds (success) DockerOperator

fetch_bcra (success) DockerOperator

dbt_build (success) DockerOperator

Read Flow

```
graph LR; A[raw_to_s3_rds] --> B[fetch_bcra]; B --> C[dbt_build]
```

VALIDACION DBT EC2

The screenshot shows the AWS CloudShell interface running a dbt validation command. The command executed was:

```
dbt validate
```

The output of the validation command is displayed in the CloudShell terminal window:

```
07:39:35 79 of 86 START test __dbt_expectations__expect_column_values_to_be_between_mart_reviews_trend_by_district_month_12_1 [OK]
07:39:35 79 of 86 START test __dbt_expectations__expect_column_values_to_be_between_mart_reviews_trend_by_district_reviews_pc_0 [OK]
07:39:35 79 of 86 PASS notNullLament_price_and_listing_neighbourhood_group .... [PASS in 0.19s]
07:39:35 80 of 86 START test __dbt_expectations__expect_column_values_to_be_between_mart_reviews_trend_by_district_year_2019_2013 [OK]
07:39:35 77 of 86 PASS uniqueLament_price_and_listing_neighbourhood_group .... [PASS in 0.16s]
07:39:35 83 of 86 START test __dbt_utils__unique_combination_of_columns_mart_reviews_trend_by_district_year_month_neighbourhood_group [OK]
07:39:35 78 of 86 PASS __dbt_expectations__expect_column_values_to_be_between_mart_reviews_trend_by_district_month_12_1 [PASS in 0.14s]
07:39:35 82 of 86 START test notNullLament_reviews_trend_by_district_neighbourhood_group [OK]
07:39:35 79 of 86 PASS __dbt_expectations__expect_column_values_to_be_between_mart_reviews_trend_by_district_reviews_pc_0 [PASS in 0.23s]
07:39:35 89 of 86 PASS __dbt_expectations__expect_column_values_to_be_between_mart_reviews_trend_by_district_year_2019_2013 [PASS in 0.18s]
07:39:35 83 of 86 START test __dbt_expectations__expect_column_values_to_be_between_mart_reviewtype_supply_and_rev_listings_0 [OK]
07:39:35 84 of 86 START test __dbt_expectations__expect_column_values_to_be_between_mart_reviewtype_supply_and_rev_revenue_est_0 [OK]
07:39:35 81 of 86 PASS __dbt_utils__unique_combination_of_columns_mart_reviews_trend_by_district_year_month_neighbourhood_group [PASS in 0.26s]
07:39:35 85 of 86 START test notNullLament_reviewtype_supply_and_rev_rev_type ..... [OK]
07:39:35 82 of 86 PASS notNullLament_reviews_trend_by_district_neighbourhood_group .... [PASS in 0.26s]
07:39:35 86 of 86 START test uniqueLament_reviewtype_supply_and_rev_rev_type ..... [OK]
07:39:36 84 of 86 PASS __dbt_expectations__expect_column_values_to_be_between_mart_reviewtype_supply_and_rev_revenue_est_0 [PASS in 0.16s]
07:39:36 83 of 86 PASS __dbt_expectations__expect_column_values_to_be_between_mart_reviewtype_supply_and_rev_listings_0 [PASS in 0.17s]
07:39:36 85 of 86 PASS notNullLament_reviewtype_supply_and_rev_rev_type ..... [PASS in 0.16s]
07:39:36 86 of 86 PASS uniqueLament_reviewtype_supply_and_rev_rev_type ..... [PASS in 0.18s]
07:39:36 
07:39:36 Finished running 9 view models, 5 table models, 72 tests in 8 hours 8 minutes and 5.59 seconds (5.59s).
07:39:36 
07:39:36 Completed successfully.
07:39:36 
07:39:36 Done. PASS=86 FAIL=0 SKIPPED=0 TOTAL=86
[Done] Pipeline complete.
Project: Integrador, Job: ETL
```