```sql
-- 05_validations.sql
-- Phase 3: Validation and data quality checks
-- Read-only checks: safe to re-run anytime.

SET search_path TO airline, public;

------------------------------------------------------------
-- 1. ROW COUNTS BY TABLE
------------------------------------------------------------

SELECT 'airlines'           AS table_name, COUNT(*) AS row_count FROM airline.airlines
UNION ALL
SELECT 'airports'           AS table_name, COUNT(*) AS row_count FROM airline.airports
UNION ALL
SELECT 'aircraft'           AS table_name, COUNT(*) AS row_count FROM airline.aircraft
UNION ALL
SELECT 'routes'             AS table_name, COUNT(*) AS row_count FROM airline.routes
UNION ALL
SELECT 'flights'            AS table_name, COUNT(*) AS row_count FROM airline.flights
UNION ALL
SELECT 'flight_performance' AS table_name, COUNT(*) AS row_count FROM
airline.flight_performance
UNION ALL
SELECT 'passengers'         AS table_name, COUNT(*) AS row_count FROM
airline.passengers
UNION ALL
SELECT 'loyalty_accounts'   AS table_name, COUNT(*) AS row_count FROM
airline.loyalty_accounts
UNION ALL
SELECT 'miles_transactions' AS table_name, COUNT(*) AS row_count FROM
airline.miles_transactions
UNION ALL
SELECT 'bookings'           AS table_name, COUNT(*) AS row_count FROM airline.bookings
UNION ALL
SELECT 'payments'           AS table_name, COUNT(*) AS row_count FROM airline.payments
ORDER BY table_name;


------------------------------------------------------------
-- 2. FOREIGN-KEY “PROBLEM COUNTS”
-- These should all be 0 if constraints + cleanup worked.
------------------------------------------------------------
```

```
WITH
flights_fk_issues AS (
   SELECT
       SUM(CASE WHEN a.airline_id IS NULL THEN 1 ELSE 0 END) AS
flights_missing_airline,
       SUM(CASE WHEN o.airport_id IS NULL THEN 1 ELSE 0 END) AS
flights_missing_origin_airport,
       SUM(CASE WHEN d.airport_id IS NULL THEN 1 ELSE 0 END) AS
flights_missing_destination_airport
   FROM airline.flights f
   LEFT JOIN airline.airlines a
       ON f.airline_id = a.airline_id
   LEFT JOIN airline.airports o
       ON f.origin_airport_id = o.airport_id
   LEFT JOIN airline.airports d
       ON f.destination_airport_id = d.airport_id
),
bookings_fk_issues AS (
   SELECT
       SUM(CASE WHEN p.passenger_id IS NULL THEN 1 ELSE 0 END) AS
bookings_missing_passenger,
       SUM(CASE WHEN fl.flight_id    IS NULL THEN 1 ELSE 0 END) AS
bookings_missing_flight
   FROM airline.bookings b
   LEFT JOIN airline.passengers p
       ON b.passenger_id = p.passenger_id
   LEFT JOIN airline.flights fl
       ON b.flight_id = fl.flight_id
),
payments_fk_issues AS (
   SELECT
       SUM(CASE WHEN b.booking_id IS NULL THEN 1 ELSE 0 END) AS
payments_missing_booking
   FROM airline.payments pay
   LEFT JOIN airline.bookings b
       ON pay.booking_id = b.booking_id
)
SELECT
   f.flights_missing_airline,
   f.flights_missing_origin_airport,
   f.flights_missing_destination_airport,
```

```sql
   b.bookings_missing_passenger,
   b.bookings_missing_flight,
   p.payments_missing_booking
FROM flights_fk_issues f
CROSS JOIN bookings_fk_issues b
CROSS JOIN payments_fk_issues p;


------------------------------------------------------------
-- 3. UNIQUENESS & BUSINESS KEY CHECKS
------------------------------------------------------------

-- Airlines: IATA and ICAO should be unique when present
SELECT
   'airlines_iata' AS check_name,
   COUNT(*) FILTER (WHERE iata_code IS NOT NULL) AS non_null_count,
   COUNT(DISTINCT iata_code) FILTER (WHERE iata_code IS NOT NULL) AS distinct_non_null
FROM airline.airlines
UNION ALL
SELECT
   'airlines_icao' AS check_name,
   COUNT(*) FILTER (WHERE icao_code IS NOT NULL) AS non_null_count,
   COUNT(DISTINCT icao_code) FILTER (WHERE icao_code IS NOT NULL) AS distinct_non_null
FROM airline.airlines
ORDER BY check_name;

-- Airports: IATA and ICAO should be unique when present
SELECT
   'airports_iata' AS check_name,
   COUNT(*) FILTER (WHERE iata_code IS NOT NULL) AS non_null_count,
   COUNT(DISTINCT iata_code) FILTER (WHERE iata_code IS NOT NULL) AS distinct_non_null
FROM airline.airports
UNION ALL
SELECT
   'airports_icao' AS check_name,
   COUNT(*) FILTER (WHERE icao_code IS NOT NULL) AS non_null_count,
   COUNT(DISTINCT icao_code) FILTER (WHERE icao_code IS NOT NULL) AS distinct_non_null
FROM airline.airports
ORDER BY check_name;

-- Passengers: email uniqueness (when present)
SELECT
```

```
   'passengers_email' AS check_name,
   COUNT(*) FILTER (WHERE email IS NOT NULL) AS non_null_count,
   COUNT(DISTINCT email) FILTER (WHERE email IS NOT NULL) AS distinct_non_null
FROM airline.passengers;

-- Loyalty: at most one loyalty account per passenger
SELECT
   'loyalty_per_passenger' AS check_name,
   COUNT(*) AS loyalty_rows,
   COUNT(DISTINCT passenger_id) AS distinct_passengers
FROM airline.loyalty_accounts;


------------------------------------------------------------
-- 4. BTS FLIGHT PERFORMANCE SANITY
-- Quick shape of BTS summary data for documentation.
------------------------------------------------------------

-- Coverage (years, months, airlines, airports)
SELECT
   MIN(year)  AS min_year,
   MAX(year)  AS max_year,
   COUNT(DISTINCT year)  AS years_covered,
   COUNT(DISTINCT month) AS months_covered,
   COUNT(DISTINCT airline_iata) AS airlines_in_bts,
   COUNT(DISTINCT airport_iata) AS airports_in_bts,
   COUNT(*) AS total_rows
FROM airline.flight_performance;

-- Delay reason totals (basic sanity)
SELECT
   SUM(arrivals)                       AS total_arrivals,
   SUM(arrivals_delayed_15min)         AS total_arrivals_15min_delayed,
   SUM(arr_cancelled)                  AS total_arr_cancelled,
   SUM(arr_diverted)                   AS total_arr_diverted,
   SUM(total_arrival_delay_min)        AS total_delay_minutes,
   SUM(carrier_delay)                  AS total_carrier_delay_min,
   SUM(weather_delay)                  AS total_weather_delay_min,
   SUM(nas_delay)                      AS total_nas_delay_min,
   SUM(security_delay)                 AS total_security_delay_min,
   SUM(late_aircraft_delay)            AS total_late_aircraft_delay_min
FROM airline.flight_performance;
```

```sql
------------------------------------------------------------
-- 5. REVENUE & BOOKINGS SANITY
------------------------------------------------------------

-- Distribution of base fares by fare_class
SELECT
   fare_class,
   COUNT(*)                    AS bookings,
   MIN(base_price_usd)         AS min_price,
   MAX(base_price_usd)         AS max_price,
   AVG(base_price_usd)         AS avg_price
FROM airline.bookings
GROUP BY fare_class
ORDER BY fare_class;

-- Payments vs base_price: simple ratio stats
SELECT
   COUNT(*) AS payment_rows,
   MIN(amount_usd) AS min_payment,
   MAX(amount_usd) AS max_payment,
   AVG(amount_usd) AS avg_payment
FROM airline.payments;

-- Compare bookings vs payments by booking_id
SELECT
   'bookings_without_payments' AS metric,
   COUNT(*) AS count_rows
FROM airline.bookings b
LEFT JOIN airline.payments p
   ON b.booking_id = p.booking_id
WHERE p.booking_id IS NULL
UNION ALL
SELECT
   'payments_without_bookings' AS metric,
   COUNT(*) AS count_rows
FROM airline.payments p
LEFT JOIN airline.bookings b
   ON p.booking_id = b.booking_id
WHERE b.booking_id IS NULL;
```

```sql
------------------------------------------------------------
-- 6. HIGH-LEVEL JOIN SANITY:
-- How many flights have at least one booking?
------------------------------------------------------------

SELECT
   COUNT(DISTINCT f.flight_id) AS flights_with_bookings,
   (SELECT COUNT(*) FROM airline.flights) AS total_flights
FROM airline.flights f
JOIN airline.bookings b
 ON f.flight_id = b.flight_id;
```