SQL Data Analyst Checklist — Daily Use (With SQL Examples)

This PDF contains a practical, copy-paste friendly checklist with SQL syntax examples for each common data profiling, cleaning, transformation, validation and pre-analysis task.

1. DATA PROFILING (Understand the data first)

- List all tables in the database/schema
```
/* MySQL / PostgreSQL */
SELECT table_name FROM information_schema.tables WHERE table_schema = 'your_schema';
```

- Check row counts for important tables
```
SELECT COUNT(*) AS row_count FROM schema.table_name;
```

- Review columns and data types
```
SELECT column_name, data_type, is_nullable
FROM information_schema.columns
WHERE table_name = 'table_name' AND table_schema = 'your_schema';
```

- Quick distinct value sample
```
SELECT column, COUNT(*) AS freq
FROM schema.table
GROUP BY column
ORDER BY freq DESC
LIMIT 20;
```

2. DATA QUALITY CHECKS

- Check NULLs in critical fields
```
SELECT COUNT(*) AS null_count FROM schema.table WHERE critical_col IS NULL;
```

- Find duplicate rows by business key
```
SELECT business_key, COUNT(*) AS cnt
FROM schema.table
GROUP BY business_key
HAVING COUNT(*) > 1;
```

- Detect invalid email formats (basic)
```
SELECT * FROM schema.table WHERE email NOT LIKE '%@%.%';
```

- Date parsing problems (strings that fail to cast)
```
/* PostgreSQL */
SELECT * FROM schema.table WHERE to_date(date_string, 'YYYY-MM-DD') IS NULL;

/* MySQL - example to find non-conforming rows */
SELECT * FROM schema.table WHERE STR_TO_DATE(date_string, '%Y-%m-%d') IS NULL;
```

3. DATA CLEANING (Fix issues)

- Replace NULLs with default or computed values
```
UPDATE schema.table
SET numeric_col = 0
WHERE numeric_col IS NULL;
```

- Standardize text (trim, lower)
```
UPDATE schema.table
SET name = TRIM(LOWER(name));
```

- Normalize phone numbers (remove non-digits)
```
/* PostgreSQL */
UPDATE schema.table
SET phone = REGEXP_REPLACE(phone, '[^0-9]', '', 'g');
```

```
/* MySQL 8+ */
UPDATE schema.table
SET phone = REGEXP_REPLACE(phone, '[^0-9]', '');
```

- Remove exact duplicate rows (keep first/lowest id)

```
DELETE FROM schema.table
WHERE id NOT IN (
  SELECT MIN(id) FROM schema.table GROUP BY business_key
);
```

4. DATA TRANSFORMATION (Prepare analysis-ready fields)

- Add derived columns (age example)

```
ALTER TABLE schema.table ADD COLUMN age INT;

UPDATE schema.table
SET age = DATE_PART('year', AGE(CURRENT_DATE, dob)); -- PostgreSQL

-- MySQL equivalent
UPDATE schema.table
SET age = TIMESTAMPDIFF(YEAR, dob, CURDATE());
```

- Extract date parts

```
SELECT order_id, order_date,
       EXTRACT(YEAR FROM order_date) AS year,
       EXTRACT(MONTH FROM order_date) AS month,
       EXTRACT(DAY FROM order_date) AS day
FROM schema.orders;
```

- Create flags and segments

```
ALTER TABLE schema.orders ADD COLUMN is_high_value BOOLEAN DEFAULT FALSE;

UPDATE schema.orders
SET is_high_value = TRUE
WHERE total_amount >= 1000;
```

5. VALIDATION (Verify correctness after cleaning)

- Re-check row counts & nulls

```
SELECT COUNT(*) FROM schema.table;
SELECT COUNT(*) FROM schema.table WHERE important_col IS NULL;
```

- Ensure referential integrity (find orphan rows)

```
SELECT c.*
FROM schema.child_table c
LEFT JOIN schema.parent_table p ON c.parent_id = p.id
WHERE p.id IS NULL;
```

- Sample-based validation (compare aggregates)

```
SELECT COUNT(*) AS before_count FROM raw.table;
SELECT COUNT(*) AS after_count FROM cleaned.table;

SELECT SUM(amount) AS before_sum FROM raw.table;
SELECT SUM(amount) AS after_sum FROM cleaned.table;
```

6. PRE-ANALYSIS CHECKS (Quick readiness tests)

- Column distributions & cardinality

```
SELECT category, COUNT(*) AS cnt
FROM schema.table
GROUP BY category
```

```
ORDER BY cnt DESC
LIMIT 50;
```

- Verify date range and completeness

```
SELECT MIN(event_date) AS start_date, MAX(event_date) AS end_date FROM schema.events;

-- Check missing dates (example for PostgreSQL using generate_series)
SELECT d::date
FROM generate_series((SELECT MIN(event_date) FROM schema.events), (SELECT MAX(event_date) FROM sche
LEFT JOIN schema.events e ON e.event_date = d::date
WHERE e.event_date IS NULL;
```

7. DOCUMENT & STORE CLEAN VERSION

- Save cleaned table (materialize)

```
CREATE TABLE schema.cleaned_table AS
SELECT * FROM schema.table_after_cleaning;
```

- Data dictionary export (example)

```
SELECT column_name, data_type, is_nullable, character_maximum_length
FROM information_schema.columns
WHERE table_schema='schema' AND table_name='cleaned_table';
```

8. START ANALYSIS & KPIs (Quick SQL snippets)

- Basic summary statistics

```
SELECT
  COUNT(*) AS total_orders,
  COUNT(DISTINCT customer_id) AS active_customers,
  SUM(total_amount) AS revenue,
  AVG(total_amount) AS avg_order_value
FROM schema.orders
WHERE order_date BETWEEN '2025-01-01' AND '2025-12-31';
```

- Top customers by revenue

```
SELECT customer_id, SUM(total_amount) AS revenue
FROM schema.orders
GROUP BY customer_id
ORDER BY revenue DESC
LIMIT 20;
```

Notes:

- Replace `schema.table` with your schema and table names. Use transactions and backups before running destructive updates/deletes.
- Adapt date functions to your SQL dialect (MySQL/Postgres/Redshift/BigQuery).