# Cheatsheet – Day 1

**Platform, ELT Ingestion, Delta Fundamentals**

---

## Unity Catalog Hierarchy

```
Metastore
  └── Catalog
        └── Schema (Database)
              ├── Table (Managed / External)
              ├── View
              ├── Function
              └── Volume (Managed / External)
```

---

## Cluster Types

| Type | Purpose |
|---|---|
| All-Purpose | Interactive dev, notebooks |
| Job (Jobs Compute) | Automated job execution |
| SQL Warehouse | SQL queries, BI dashboards |

**Access Mode:** Single User (full libs) vs Shared (governed, Unity Catalog enforced)

---

## Reading Files

```sql
-- CSV with explicit schema
SELECT * FROM read_files('/path/to.csv',
    format => 'csv',
    header => 'true',
    sep => ',');

-- JSON
SELECT * FROM read_files('/path/to.json', format => 'json');
```

```python
df = spark.read.format("csv") \
    .option("header", "true") \
    .option("inferSchema", "true") \
    .load("/path/to.csv")
```

---

## Creating Tables

```sql
-- Managed table (data managed by Unity Catalog)
CREATE TABLE catalog.schema.my_table
AS SELECT * FROM read_files('/path', format => 'csv', header
        => 'true');

-- External table (data at external location, survives DROP)
CREATE TABLE catalog.schema.ext_table
LOCATION 's3://bucket/path'
AS SELECT * FROM read_files('/path', format => 'csv', header
        => 'true');
```

| | Managed Table | External Table |
|---|---|---|
| Data location | Controlled by UC | User-specified LOCATION |
| DROP TABLE | Deletes metadata + data | Deletes metadata ONLY |
| Best for | Most use cases | Shared storage, legacy data |

---

## Schema Evolution

```
# Auto-merge new columns during write
(df.writeStream
    .option("mergeSchema", "true")   # add new columns
    .toTable("target"))

# Overwrite entire schema (replace all columns)
(df.write.mode("overwrite")
    .option("overwriteSchema", "true")
    .saveAsTable("target"))
```

| Option | Behavior |
|---|---|
| `mergeSchema` | Adds new columns, keeps existing |
| `overwriteSchema` | Replaces schema entirely |

---

## Magic Commands

| Command | Purpose | Example |
|---|---|---|
| `%sql` | Run SQL in Python notebook | `%sql SELECT * FROM t` |
| `%python` | Run Python in SQL notebook | `%python print("hello")` |
| `%run` | Execute another notebook | `%run ./setup_notebook` |
| `%pip` | Install Python packages | `%pip install pandas` |
| `%fs` | DBFS file operations | `%fs ls /Volumes/...` |

---

## Temp Views & SQL

```
df.createOrReplaceTempView("my_view")
result = spark.sql("SELECT * FROM my_view WHERE col > 10")
```

---

## Common Transformations

```
df.select("col1", "col2")
df.filter(col("price") > 0)
df.withColumn("total", col("qty") * col("price"))
df.withColumnRenamed("old", "new")
df.drop("unwanted_col")
df.dropDuplicates(["id"])
df.na.fill({"col": "default"})
```

---

## Delta Lake Basics

- **Format:** Parquet + JSON transaction log ( `_delta_log/` )
- **ACID:** Atomic, Consistent, Isolated, Durable
- **Default:** All tables in Databricks are Delta by default

```
DESCRIBE HISTORY my_table;
SELECT * FROM my_table VERSION AS OF 3;
SELECT * FROM my_table TIMESTAMP AS OF '2025-01-01';
RESTORE TABLE my_table TO VERSION AS OF 3;
```

---

## Key Exam Points

- Unity Catalog is **deny-by-default**
- `USE CATALOG` + `USE SCHEMA` required before `SELECT`
- Managed table: data deleted on DROP
- External table: data survives DROP
- Volumes: manage non-tabular files (CSV, JSON, images)
- Delta transaction log: one JSON file per commit
- `mergeSchema` = additive (new columns); `overwriteSchema` = destructive (replace)
- `%run` executes another notebook in same context (shares variables)

- ELT = Extract → Load raw → Transform in-place (Databricks model)
- ETL = Extract → Transform outside → Load (traditional)