Data Engineer Cheat Sheet - Expanded (Interview Ready)

Incremental Load (Delta)

```
SQL:
SELECT * FROM Orders WHERE LastModifiedDate > (SELECT MAX(LastLoadDate) FROM LoadTracking);
Python:
last_load = '2025-09-01'
df = pd.read_sql(f"SELECT * FROM Orders WHERE LastModifiedDate > '{last_load}'", conn)
PySpark:
last_load = '2025-09-01'
df = spark.read.format('delta').load('/mnt/bronze/orders')
filtered = df.filter(df['LastModifiedDate'] > last_load)
```

SCD Type 2 (Slowly Changing Dimension)

```
SQL:
MERGE INTO DimCustomer t
USING StgCustomer s
ON t.CustomerID = s.CustomerID AND t.IsCurrent = 1
WHEN MATCHED AND (t.Name <> s.Name OR t.Address <> s.Address)
  THEN UPDATE SET t.IsCurrent = 0, t.ValidTo = GETDATE()
WHEN NOT MATCHED BY TARGET
  THEN INSERT (CustomerID, Name, Address, ValidFrom, IsCurrent)
       VALUES (s.CustomerID, s.Name, s.Address, GETDATE(), 1);
PySpark:
stg = spark.read.parquet('/mnt/silver/stg_customer')
dim = spark.read.format('delta').load('/mnt/gold/dim_customer')
# Join and find changes
changed = stg.join(dim.filter(dim.IsCurrent == 1), 'CustomerID', 'left')\
              .filter((stg.Name != dim.Name) | (stg.Address != dim.Address))
# Mark old record inactive
updates = dim.join(changed, 'CustomerID').withColumn('IsCurrent', F.lit(0))
# Insert new record
new_records = stg.join(changed, 'CustomerID').withColumn('IsCurrent', F.lit(1))
```

Window Functions

```
SQL:
SELECT EmployeeID, Salary, RANK() OVER (PARTITION BY Department ORDER BY Salary DESC) AS SalaryRank FROM E
PySpark:
from pyspark.sql.window import Window
windowSpec = Window.partitionBy('Department').orderBy(F.desc('Salary'))
df = df.withColumn('SalaryRank', F.rank().over(windowSpec))
```

Joins Optimization

```
Broadcast Join (small + large table):
df_joined = big_df.join(broadcast(small_df), 'id')
Skew Handling:
big_df = big_df.repartition(200, 'id')
```

GDPR / Masking Example

```
SQL Dynamic Masking:
CREATE TABLE Customer( SSN VARCHAR(20) MASKED WITH (FUNCTION='partial(0,"XXX-XX-",4)') );
PySpark PII Masking:
df = df.withColumn('masked_email', F.regexp_replace('email', '(^[^@]{3}|(?!^)\G)[^@]', '*'))
```