Databricks 14 Days AI Challenge

At Indian Data Club, we're on a mission to empower the next generation with practical AI and data skills. Our **Databricks 14-Days AI Challenge** is designed to help beginners build a strong foundation Databricks through daily learning, hands-on practice, and real-world problem solving.

This Challenge is Sponsored by Databricks and organised by **Codebasics** and Indian Data Club.

▼ Challenge Structure (Two Phases)

Phase 1 : Learning & Sharing (9th Jan – 22nd Jan) By **Indian Data Club**

- 14 days of hands-on learning
- Understand Databricks concepts and workflows
- Community-driven learning and sharing in Socials

Phase 2 : Project Phase (24th Jan – 30th Jan) By **Codebasics**

(Once the project will live on Codebasics website we will send an email to register for the project)

- 7 days of project work with Problem Statement
- Apply what you've learned
- Earn Rewards

▼ 📣 Announcement : (Check daily)

The challenge will start on 9th **January**.
Before that, join our **Online Induction Session** to clear doubts and get a walkthrough of the Notion workspace.

Date: 8th January

Time: 7:00 PM (IST)

Zoom Link: Click here to join

During the induction, you will get to know:

1. What is the Databricks AI Challenge
2. Why it is important to learn
3. A short introduction to Databricks by **Ashish Mishra : LinkedIn**
4. Step-by-step guidance on: [By **Sahib** : LinkedIn]
    - How to learn daily and post on social media
    - How to submit the challenge
    - How to support your peers on Discord

🗺️ How to Own It

- **Keep Your Streak Lit** 🔥

  Share your daily progress on X (Twitter), LinkedIn, and Instagram Reels. Tag **@Databricks, @Codebasics** and **@Indiandataclub** for a shoutout,

  and add the hashtag `#DatabricksWithIDC`. Stay consistent, and you might just hit IDC's **Hall of Fame**!

- **Support From Us: Join Discord (Click Here)**
    - For daily updates, use the IDC-Challenges channel.
    - To ask questions, use the Challenges Doubt channel.
    - To share your post link, use the Challenge Submissions channel.
    - You can also join IDC Voice General anytime to connect with peers.
- **Need Help?**

  If you have any doubts, tag **@databricksbuddies** on Discord for assistance.

🦾 Most Important: Submission Guidelines (Need to Submit daily)

To maintain your 14-days streak, please follow these steps:

- ✅ **Submit your daily progress using this form:**

  (Please complete this form daily after posting your progress.)

  Each day's challenge form is also available in the roadmap under the *After Daily Challenge* section.

📣**Share your learning and build your online presence:**

1. Post your daily learning on **LinkedIn, X (Twitter), Instagram (Reels), or GitHub**, tagging **@Databricks, @Codebasics** and **@Indiandataclub** and add the hashtag `#DatabricksWithIDC`.
2. Fill out the submission form with a few quick details (Daily): https://forms.office.com/r/GT2BquTRnH
3. Download this Personal Tracker and use it for 14 Days: Download (Click Here)

Note: Start by following the Notion plan daily. The structure is flexible and may be updated based on requirements. P

lease make sure to follow the plan sequentially, and **bookmark this Notion roadmap** for quick and easy access.

Here is the link to sign up for Databricks: Click Here

Watch this tutorial to learn about Databricks and its features:
https://youtu.be/761SQ9Hxbic?si=J7K2CCbe7_VOzZo0

▼ DAY 0 – Setup & Data Loading (Prerequisites)

Overview

Before starting Day 1, complete this setup to load the e-commerce dataset directly from Kaggle into your Databricks workspace.

▼ Step 1: Create Databricks Account

1. Go to Databricks Community Edition
2. Sign up for free account
3. Verify email and log in

4. Create a cluster (use default settings)

▼ Step 2: Get Kaggle API Credentials

1. Go to Kaggle.com and log in
2. Click on your profile picture → **Account**
3. Scroll to **API** section → Click **Create New API Token**
4. Download `kaggle.json` (contains your credentials)
5. Open the file and note your `username` and `key`

▼ Step 3: Load Data in Databricks

Create a new notebook in Databricks and run these cells:

▼ Notebook:

▼ 1. Install Dependencies

```
!pip install kaggle
```

▼ 2. Configure Kaggle Credentials

```
import os

os.environ["KAGGLE_USERNAME"] = "your_username"
os.environ["KAGGLE_KEY"] = "your_key"

print("Kaggle credentials configured!")
```

▼ 3. Create Database Schema

```
spark.sql("""
CREATE SCHEMA IF NOT EXISTS workspace.ecommerce
""")
```

▼ 4. Create Volume for Data Storage

```
spark.sql("""
CREATE VOLUME IF NOT EXISTS workspace.ecommerce.ecom
merce_data
""")
```

▼ 5. Download Dataset from Kaggle

```
cd /Volumes/workspace/ecommerce/ecommerce_data
kaggle datasets download -d mkechinov/ecommerce-behavior-
data-from-multi-category-store
```

▼ 6. Extract Downloaded Dataset

```
cd /Volumes/workspace/ecommerce/ecommerce_data
unzip -o ecommerce-behavior-data-from-multi-category-store.
zip
ls -lh
```

▼ 7. Clean Up Zip File

```
cd /Volumes/workspace/ecommerce/ecommerce_data
rm -f ecommerce-behavior-data-from-multi-category-store.zip
ls -lh
```

▼ 8. Restart Python Environment

```
%restart_python
```

▼ 9. Load November 2019 Data

```
df_n = spark.read.csv("/Volumes/workspace/ecommerce/ecom
merce_data/2019-Nov.csv")
```

▼ 10. Load October 2019 Data

```
df = spark.read.csv("/Volumes/workspace/ecommerce/ecommerce_data/2019-Oct.csv")
```

▼ 11. Display Dataset Statistics and Schema

```
print(f"October 2019 - Total Events: {df.count():,}")
print("\n" + "="*60)
print("SCHEMA:")
print("="*60)
df.printSchema()
```

▼ 12. Display Sample Data

```
print("\n" + "="*60)
print("SAMPLE DATA (First 5 rows):")
print("="*60)
df.show(5, truncate=False)
```

For better understanding, check the setup guide video: https://youtu.be/nHGMcrxHqrA

▼ Expected Schema

After loading, your dataset will have **9 columns**:

Column	Type	Description	Notes
event_time	timestamp	When event happened (UTC)	Format: YYYY-MM-DD HH:MM:SS UTC
event_type	string	Type of event	Values: view, cart, purchase, remove_from_cart
product_id	long	Unique product identifier	Numeric ID
category_id	long	Category identifier	Numeric ID

Column	Type	Description	Notes
category_code	string	Category hierarchy	e.g., "electronics.smartphone" (can be null)
brand	string	Product brand	Lowercase (can be null)
price	double	Product price in USD	Positive values
user_id	long	Permanent user identifier	Numeric ID
user_session	string	Session identifier	UUID format, changes per session

▼ Dataset Size Reference

File	Events	Size	Recommended Use
2019-Oct.csv	~4.2M	~1.1 GB	Days 1-3 (learning basics)
2019-Nov.csv	~9.3M	~2.2 GB	Days 4+ (full analysis)
Combined	~13.5M	~3.3 GB	Days 8+ (comprehensive analysis)

💡 **Pro Tip:** Start with October data for faster iterations, then scale to combined dataset

▼ Troubleshooting

▼ Issue: Kaggle credentials not working

```
# Method 1: Set credentials directly
os.environ['KAGGLE_USERNAME'] = "your_username"
os.environ['KAGGLE_KEY'] = "your_key"

# Method 2: Verify kaggle config
!cat ~/.kaggle/kaggle.json
```

▼ Issue: File not found in DBFS

```
# Check DBFS contents
display(dbutils.fs.ls("/FileStore/ecommerce_data/"))

# Alternative path check
dbutils.fs.ls("dbfs:/FileStore/ecommerce_data/")
```

▼ **Issue: Memory errors with large dataset**

```
# Use sampling for testing
sampled = load_ecommerce_data("Oct", sample_fraction=0.1)
print(f"Sampled data: {sampled.count():,} rows")
```

▼ **Issue: Schema inference problems**

```
# Explicitly define schema
from pyspark.sql.types import StructType, StructField, Timesta
mpType, StringType, LongType, DoubleType

schema = StructType([
    StructField("event_time", TimestampType(), True),
    StructField("event_type", StringType(), True),
    StructField("product_id", LongType(), True),
    StructField("category_id", LongType(), True),
    StructField("category_code", StringType(), True),
    StructField("brand", StringType(), True),
    StructField("price", DoubleType(), True),
    StructField("user_id", LongType(), True),
    StructField("user_session", StringType(), True)
])

events = spark.read.csv("/FileStore/ecommerce_data/2019-Oct.
csv",
                        header=True, schema=schema)
```

▼ Alternative: Manual Upload (If Kaggle API Doesn't Work)

1. **Download manually:**
    - Go to Kaggle Dataset
    - Click "Download" button
    - Extract `2019-Oct.csv` and `2019-Nov.csv`
2. **Upload to Databricks:**
    - Method A: Click **Data** → **Create Table** → Upload files
    - Method B: Drag and drop files into notebook cell
3. **Note the path:**

```
# Path will be shown after upload, typically:
# /FileStore/tables/2019_Oct.csv

events = spark.read.csv("/FileStore/tables/2019_Oct.csv",
                        header=True, inferSchema=True)
```

✅ Setup Complete Checklist

- ☑ ~~Databricks account created and cluster running~~
- ☑ ~~Kaggle credentials configured~~
- ☑ ~~Dataset downloaded (2019-Oct.csv, 2019-Nov.csv)~~
- ☑ ~~Files uploaded to DBFS~~
- ☑ ~~Data loaded successfully (verified with count)~~
- ☑ ~~Schema validated (9 columns confirmed)~~
- ☑ ~~Sample queries executed successfully~~

🚀 Ready for Day 1!

Once all checklist items are complete, you're ready to start **Day 1: Platform Setup & First Steps**

Quick Start Code for Day 1:

```
# Load your data
events = load_ecommerce_data("Oct")

# Verify it's working
print(f"✅ Ready to go! Loaded {events.count():,} events")
events.show(5)

# Your Day 1 challenges start here...
```

📚 Additional Resources

- Kaggle Dataset Page
- Databricks Documentation
- PySpark DataFrame Guide
- Dataset provided by REES46 Open CDP

Daily Challenge:

▼ PHASE 1: FOUNDATION (Days 1-4)

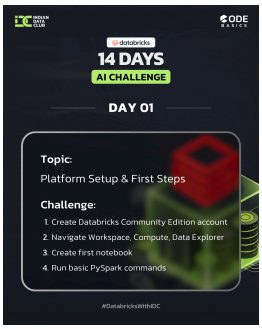
▼ DAY 1 – Platform Setup & First Steps

Learn:

- Why Databricks vs Pandas/Hadoop?
- Lakehouse architecture basics
- Databricks workspace structure
- Industry use cases (Netflix, Shell, Comcast)

🛠️ Tasks:

1. Create Databricks Community Edition account
2. Navigate Workspace, Compute, Data Explorer
3. Create first notebook
4. Run basic PySpark commands



📝 Practice:

```
# Create simple DataFrame
data = [("iPhone", 999), ("Samsung", 799), ("MacBook", 1299)]
df = spark.createDataFrame(data, ["product", "price"])
df.show()

# Filter expensive products
df.filter(df.price > 1000).show()
```

🔗 Resources:

- Databricks Trial
- Quickstart Guide

▼ ✅ **Submit your day 1 progress using this form:**

1. Share your learning and build online presence.

Post your daily learning on **LinkedIn, X (Twitter), Instagram (Reels), or GitHub tagging @Databricks, @Codebasics and @indiandataclub along with #DatabricksWithIDC.**

2. Fill the submission form: https://forms.office.com/r/GT2BquTRnH

▼ DAY 2 – Apache Spark Fundamentals

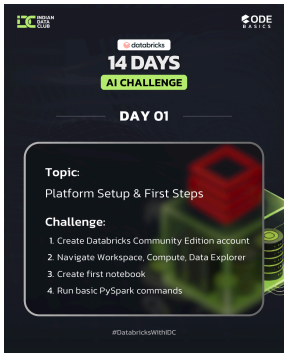
Learn:

- Spark architecture (driver, executors, DAG)
- DataFrames vs RDDs

- Lazy evaluation
- Notebook magic commands (`%sql` , `%python` , `%fs`)

🛠️ Tasks:

1. Upload sample e-commerce CSV
2. Read data into DataFrame
3. Perform basic operations: select, filter, groupBy, orderBy
4. Export results



📝 Practice:

```
# Load data
events = spark.read.csv("/path/to/sample.csv", header=True, inferSchema
=True)

# Basic operations
events.select("event_type", "product_name", "price").show(10)
events.filter("price > 100").count()
events.groupBy("event_type").count().show()
top_brands = events.groupBy("brand").count().orderBy("count", ascendin
g=False).limit(5)
```

🔗 Resources:

- PySpark Guide
- Spark SQL Guide

▼ ✅ **Submit your day 2 progress using this form:**

- **1. Share your learning and build online presence.**

  Post your daily learning on **LinkedIn, X (Twitter), Instagram (Reels), or GitHub tagging @Databricks, @Codebasics and @indiandataclub along with #DatabricksWithIDC.**

  2. Fill the submission form: https://forms.office.com/r/GT2BquTRnH

▼ DAY 3 – PySpark Transformations Deep Dive

Learn:

- PySpark vs Pandas comparison
- Joins (inner, left, right, outer)
- Window functions (running totals, rankings)
- User-Defined Functions (UDFs)

🛠️ Tasks:

1. Load full e-commerce dataset
2. Perform complex joins
3. Calculate running totals with window functions
4. Create derived features

📝 Practice:

```
from pyspark.sql import functions as F
from pyspark.sql.window import Window

# Top 5 products by revenue
revenue = events.filter(F.col("event_type") == "purchase") \
    .groupBy("product_id", "product_name") \
    .agg(F.sum("price").alias("revenue")) \
    .orderBy(F.desc("revenue")).limit(5)

# Running total per user
```

```
window = Window.partitionBy("user_id").orderBy("event_time")
events.withColumn("cumulative_events", F.count("*").over(window))

# Conversion rate by category
events.groupBy("category_code", "event_type").count() \
    .pivot("event_type").sum("count") \
    .withColumn("conversion_rate", F.col("purchase")/F.col("view")*100)
```

🔗 Resources:

- Window Functions
- PySpark Functions API

▼ ✅ **Submit your day 3 progress using this form:**

Post your daily learning on **LinkedIn, X (Twitter), Instagram (Reels), or GitHub tagging @Databricks, @Codebasics and @indiandataclub along with #DatabricksWithIDC.**

2. Fill the submission form: https://forms.office.com/r/GT2BquTRnH ****

▼ DAY 4 – Delta Lake Introduction

Learn:

- What is Delta Lake?
- ACID transactions
- Schema enforcement
- Delta vs Parquet

🛠️ Tasks:

1. Convert CSV to Delta format
2. Create Delta tables (SQL and PySpark)
3. Test schema enforcement
4. Handle duplicate inserts

📝 Practice:

```
# Convert to Delta
events.write.format("delta").mode("overwrite").save("/delta/events")

# Create managed table
events.write.format("delta").saveAsTable("events_table")

# SQL approach
spark.sql("""
   CREATE TABLE events_delta
   USING DELTA
   AS SELECT * FROM events_table
""")

# Test schema enforcement
try:
   wrong_schema = spark.createDataFrame([("a","b","c")], ["x","y","z"])
   wrong_schema.write.format("delta").mode("append").save("/delta/even
ts")
except Exception as e:
   print(f"Schema enforcement: {e}")
```

🔗 Resources:

- Delta Lake Docs
- Delta Tutorial

▼ ✅ **Submit your day 4 progress using this form:**

Post your daily learning on **LinkedIn, X (Twitter), Instagram (Reels), or GitHub tagging @Databricks, @Codebasics and @indiandataclub along with #DatabricksWithIDC.**

2. Fill the submission form: https://forms.office.com/r/GT2BquTRnH

▼ PHASE 2: DATA ENGINEERING (Days 5-8)

▼ DAY 5 – Delta Lake Advanced

Learn:

- Time travel (version history)
- MERGE operations (upserts)
- OPTIMIZE & ZORDER
- VACUUM for cleanup

🛠️ Tasks:

1. Implement incremental MERGE
2. Query historical versions
3. Optimize tables
4. Clean old files

📝 Practice:

```
from delta.tables import DeltaTable

# MERGE for incremental updates
deltaTable = DeltaTable.forPath(spark, "/delta/events")
updates = spark.read.csv("/path/to/new_data.csv", header=True, inferSch
ema=True)

deltaTable.alias("t").merge(
    updates.alias("s"),
    "t.user_session = s.user_session AND t.event_time = s.event_time"
).whenMatchedUpdateAll() \
 .whenNotMatchedInsertAll() \
 .execute()

# Time travel
v0 = spark.read.format("delta").option("versionAsOf", 0).load("/delta/even
ts")
yesterday = spark.read.format("delta") \
    .option("timestampAsOf", "2024-01-01").load("/delta/events")

# Optimize
```

```
spark.sql("OPTIMIZE events_table ZORDER BY (event_type, user_id)")
spark.sql("VACUUM events_table RETAIN 168 HOURS")
```

🔗 Resources:

- Time Travel
- MERGE Guide

▼ DAY 6 – Medallion Architecture

Learn:

- Bronze (raw) → Silver (cleaned) → Gold (aggregated)
- Best practices for each layer
- Incremental processing patterns

🛠️ Tasks:

1. Design 3-layer architecture
2. Build Bronze: raw ingestion
3. Build Silver: cleaning & validation
4. Build Gold: business aggregates

📝 Practice:

```
# BRONZE: Raw ingestion
raw = spark.read.csv("/raw/events.csv", header=True, inferSchema=True)
raw.withColumn("ingestion_ts", F.current_timestamp()) \
   .write.format("delta").mode("overwrite").save("/delta/bronze/events")

# SILVER: Cleaned data
bronze = spark.read.format("delta").load("/delta/bronze/events")
silver = bronze.filter(F.col("price") > 0) \
    .filter(F.col("price") < 10000) \
    .dropDuplicates(["user_session", "event_time"]) \
    .withColumn("event_date", F.to_date("event_time")) \
```

```
    .withColumn("price_tier",
        F.when(F.col("price") < 10, "budget")
         .when(F.col("price") < 50, "mid")
         .otherwise("premium"))
silver.write.format("delta").mode("overwrite").save("/delta/silver/events")

# GOLD: Aggregates
silver = spark.read.format("delta").load("/delta/silver/events")
product_perf = silver.groupBy("product_id", "product_name") \
    .agg(
        F.countDistinct(F.when(F.col("event_type")=="view", "user_id")).alias
("views"),
        F.countDistinct(F.when(F.col("event_type")=="purchase", "user_id")).
alias("purchases"),
        F.sum(F.when(F.col("event_type")=="purchase", "price")).alias("reven
ue")
    ).withColumn("conversion_rate", F.col("purchases")/F.col("views")*100)
product_perf.write.format("delta").mode("overwrite").save("/delta/gold/pr
oducts")
```

🔗 Resources:

- Medallion Architecture
- Architecture Video

▼ DAY 7 – Workflows & Job Orchestration

Learn:

- Databricks Jobs vs notebooks
- Multi-task workflows
- Parameters & scheduling
- Error handling

🛠️ Tasks:

1. Add parameter widgets to notebooks

2. Create multi-task job (Bronze→Silver→Gold)
3. Set up dependencies
4. Schedule execution

📝 Practice:

```
# Add widgets for parameters
dbutils.widgets.text("source_path", "/default/path")
dbutils.widgets.dropdown("layer", "bronze", ["bronze","silver","gold"])

# Use parameters
source = dbutils.widgets.get("source_path")
layer = dbutils.widgets.get("layer")

def run_layer(layer_name):
    if layer_name == "bronze":
        # Bronze logic
        pass
    elif layer_name == "silver":
        # Silver logic
        pass
    # ...

# UI: Create Job
# Task 1: bronze_layer (notebook)
# Task 2: silver_layer (depends on Task 1)
# Task 3: gold_layer (depends on Task 2)
# Schedule: Daily 2 AM
```

🔗 Resources:

- Jobs Documentation
- Job Parameters

▼ DAY 8 – Unity Catalog Governance

Learn:

- Catalog → Schema → Table hierarchy
- Access control (GRANT/REVOKE)
- Data lineage
- Managed vs external tables

🛠️ Tasks:

1. Create catalog & schemas
2. Register Delta tables
3. Set up permissions
4. Create views for controlled access

📝 Practice:

```
-- Create structure
CREATE CATALOG ecommerce;
USE CATALOG ecommerce;
CREATE SCHEMA bronze;
CREATE SCHEMA silver;
CREATE SCHEMA gold;

-- Register tables
CREATE TABLE bronze.events USING DELTA LOCATION '/delta/bronze/eve
nts';
CREATE TABLE silver.events USING DELTA LOCATION '/delta/silver/event
s';
CREATE TABLE gold.products USING DELTA LOCATION '/delta/gold/produ
cts';

-- Permissions
GRANT SELECT ON TABLE gold.products TO `analysts@company.com`;
GRANT ALL PRIVILEGES ON SCHEMA silver TO `engineers@company.com
`;
```

```
-- Controlled view
CREATE VIEW gold.top_products AS
SELECT product_name, revenue, conversion_rate
FROM gold.products
WHERE purchases > 10
ORDER BY revenue DESC LIMIT 100;
```

🔗 Resources:

- Unity Catalog
- Getting Started

▼ PHASE 3: ADVANCED ANALYTICS (Days 9-11)

▼ DAY 9 – SQL Analytics & Dashboards

Learn:

- SQL warehouses
- Complex analytical queries
- Dashboard creation
- Visualizations & filters

🛠️ Tasks:

1. Create SQL warehouse
2. Write analytical queries
3. Build dashboard: revenue trends, funnels, top products
4. Add filters & schedule refresh

📝 Practice:

```
-- Revenue with 7-day moving average
WITH daily AS (
  SELECT event_date, SUM(revenue) as rev
  FROM gold.products GROUP BY event_date
)
```

```
SELECT event_date, rev,
  AVG(rev) OVER (ORDER BY event_date ROWS BETWEEN 6 PRECEDING A
ND CURRENT ROW) as ma7
FROM daily;

-- Conversion funnel
SELECT category_code,
  SUM(views) as views,
  SUM(purchases) as purchases,
  ROUND(SUM(purchases)*100.0/SUM(views), 2) as conversion_rate
FROM gold.products
GROUP BY category_code;

-- Customer tiers
SELECT
  CASE WHEN cnt >= 10 THEN 'VIP'
       WHEN cnt >= 5 THEN 'Loyal'
       ELSE 'Regular' END as tier,
  COUNT(*) as customers,
  AVG(total_spent) as avg_ltv
FROM (SELECT user_id, COUNT(*) cnt, SUM(price) total_spent
      FROM silver.events WHERE event_type='purchase' GROUP BY user_id)
GROUP BY tier;
```

🔗 Resources:

- Databricks SQL
- Dashboards Guide

▼ DAY 10 – Performance Optimization

Learn:

- Query execution plans
- Partitioning strategies
- OPTIMIZE & ZORDER

- Caching techniques

🛠️ Tasks:

1. Analyze query plans
2. Partition large tables
3. Apply ZORDER
4. Benchmark improvements

📝 Practice:

```
# Explain query
spark.sql("SELECT * FROM silver.events WHERE event_type='purchase'").
explain(True)

# Partitioned table
spark.sql("""
  CREATE TABLE silver.events_part
  USING DELTA
  PARTITIONED BY (event_date, event_type)
  AS SELECT * FROM silver.events
""")

# Optimize
spark.sql("OPTIMIZE silver.events_part ZORDER BY (user_id, product_id)")

# Benchmark
import time
start = time.time()
spark.sql("SELECT * FROM silver.events WHERE user_id=12345").count()
print(f"Time: {time.time()-start:.2f}s")

# Cache for iterative queries
cached = spark.table("silver.events").cache()
cached.count()  # Materialize
```

🔗 Resources:

- Performance Tuning
- Optimization Guide

▼ DAY 11 – Statistical Analysis & ML Prep

Learn:

- Descriptive statistics
- Hypothesis testing
- A/B test design
- Feature engineering

🛠️ Tasks:

1. Calculate statistical summaries
2. Test hypotheses (weekday vs weekend)
3. Identify correlations
4. Engineer features for ML

📝 Practice:

```
# Descriptive stats
events.describe(["price"]).show()

# Hypothesis: weekday vs weekend conversion
weekday = events.withColumn("is_weekend",
    F.dayofweek("event_date").isin([1,7]))
weekday.groupBy("is_weekend", "event_type").count().show()

# Correlation
events.stat.corr("price", "conversion_rate")

# Feature engineering
features = events.withColumn("hour", F.hour("event_time")) \
```

```
    .withColumn("day_of_week", F.dayofweek("event_date")) \
    .withColumn("price_log", F.log(F.col("price")+1)) \
    .withColumn("time_since_first_view",
        F.unix_timestamp("event_time") -
        F.first("event_time").over(Window.partitionBy("user_id").orderBy("eve
nt_time")))
```

🔗 Resources:

- Spark ML Guide

▼ PHASE 4: AI & ML (Days 12-14)

▼ DAY 12 – MLflow Basics

Learn:

- MLflow components (tracking, registry, models)
- Experiment tracking
- Model logging
- MLflow UI

🛠️ Tasks:

1. Train simple regression model
2. Log parameters, metrics, model
3. View in MLflow UI
4. Compare runs

📝 Practice:

```
import mlflow
import mlflow.sklearn
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Prepare data
```

```
df = spark.table("gold.products").toPandas()
X = df[["views", "cart_adds"]]
y = df["purchases"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# MLflow experiment
with mlflow.start_run(run_name="linear_regression_v1"):
    # Log parameters
    mlflow.log_param("model_type", "LinearRegression")
    mlflow.log_param("test_size", 0.2)

    # Train
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Evaluate
    score = model.score(X_test, y_test)
    mlflow.log_metric("r2_score", score)

    # Log model
    mlflow.sklearn.log_model(model, "model")

print(f"R² Score: {score:.4f}")
```

🔗 Resources:

- MLflow Documentation
- Model Registry

▼ DAY 13 – Model Comparison & Feature Engineering

Learn:

- Training multiple models
- Hyperparameter tuning
- Feature importance

- Spark ML Pipelines

🛠️ Tasks:

1. Train 3 different models
2. Compare metrics in MLflow
3. Build Spark ML pipeline
4. Select best model

📝 Practice:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor

models = {
    "linear": LinearRegression(),
    "decision_tree": DecisionTreeRegressor(max_depth=5),
    "random_forest": RandomForestRegressor(n_estimators=100)
}

for name, model in models.items():
    with mlflow.start_run(run_name=f"{name}_model"):
        mlflow.log_param("model_type", name)

        model.fit(X_train, y_train)
        score = model.score(X_test, y_test)

        mlflow.log_metric("r2_score", score)
        mlflow.sklearn.log_model(model, "model")

        print(f"{name}: R² = {score:.4f}")

# Spark ML Pipeline
from pyspark.ml import Pipeline
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegression as SparkLR
```

```
assembler = VectorAssembler(inputCols=["views","cart_adds"], outputCol
="features")
lr = SparkLR(featuresCol="features", labelCol="purchases")
pipeline = Pipeline(stages=[assembler, lr])

spark_df = spark.table("gold.products")
train, test = spark_df.randomSplit([0.8, 0.2])
model = pipeline.fit(train)
```

🔗 Resources:

- Spark ML

▼ DAY 14 – AI-Powered Analytics: Genie & Mosaic AI

Learn:

- Databricks Genie (natural language → SQL)
- Mosaic AI capabilities
- Generative AI integration
- AI-assisted analysis

🛠️ Tasks:

1. Use Genie to query data with natural language
2. Explore Mosaic AI features
3. Build simple NLP task
4. Create AI-powered insights

📝 Practice:

Genie Queries:

- "Show me total revenue by category"
- "Which products have the highest conversion rate?"
- "What's the trend of daily purchases over time?"

- "Find customers who viewed but never purchased"

Mosaic AI Exploration:

```
# Simple sentiment analysis or text classification
from transformers import pipeline

# Example: Analyze product review sentiment
classifier = pipeline("sentiment-analysis")
reviews = ["This product is amazing!", "Terrible quality, waste of money"]
results = classifier(reviews)

# Log to MLflow
with mlflow.start_run(run_name="sentiment_model"):
    mlflow.log_param("model", "distilbert-sentiment")
    mlflow.log_metric("accuracy", 0.95)  # Example metric
```

🔗 Resources:

- Databricks Genie
- Mosaic AI

▼ 🚀 Getting Started

1. **Download Dataset:** Kaggle Link
2. **Create Account:** Databricks Community Edition
3. **Upload Data:** DBFS or workspace
4. **Start Day 1!**

▼ Recommended Cluster:

- Runtime: 14.0+
- Node: i3.xlarge or similar
- Workers: 4-8

▼ 📚 Additional Resources

- Databricks Documentation
- PySpark API
- Delta Lake
- MLflow
- Kaggle Dataset

▼ 🏆 DAYS 15-21: CAPSTONE PROJECT WEEK

Overview

After mastering the fundamentals, it's time to build something real. This week is completely self-directed—you choose the problem, find the dataset, and architect the solution. This is your opportunity to showcase everything you've learned and create a portfolio-worthy project.

🎯 Project Guidelines

What Makes a Great Capstone?

1. **End-to-End Pipeline**: From raw data ingestion to actionable insights
2. **Real Business Value**: Solves an actual problem with measurable impact
3. **Technical Depth**: Demonstrates mastery of multiple Databricks concepts
4. **Production-Ready**: Well-documented, reproducible, and scalable
5. **Portfolio Quality**: Something you're proud to show employers

Minimum Requirements

Your capstone should include:

- ✅ **Data Architecture**: Medallion (Bronze → Silver → Gold) or similar
- ✅ **Delta Lake**: Tables with ACID transactions and optimization
- ✅ **Transformations**: Complex PySpark/SQL logic with business rules
- ✅ **Orchestration**: Automated workflow with Databricks Jobs
- ✅ **Governance**: Unity Catalog setup with proper permissions

- ✅ **Analytics**: SQL queries and/or dashboard for insights
- ✅ **ML Component** (optional but recommended): Model training with MLflow
- ✅ **Documentation**: README with architecture, setup, and findings

---

💡 Project Ideas (Not Limited To These!)

🛒 Domain: E-Commerce & Retail

1. Customer Churn Prediction System

- **Dataset**: Online Retail Dataset or Instacart Orders
- **Goals**: Build RFM model, predict churn probability, recommend retention strategies
- **Key Features**: Customer segmentation, time-series analysis, ML classification
- **Business Value**: Identify at-risk customers before they churn

2. Dynamic Pricing & Demand Forecasting

- **Dataset**: Amazon Product Reviews or Walmart Sales
- **Goals**: Forecast demand, optimize pricing, identify seasonality
- **Key Features**: Time series forecasting, price elasticity, inventory optimization
- **Business Value**: Maximize revenue through smart pricing

3. Product Recommendation Engine

- **Dataset**: Amazon Reviews or Retail Transactions
- **Goals**: Build collaborative filtering, product association rules
- **Key Features**: Market basket analysis, user-product matrix, similarity scoring
- **Business Value**: Increase cross-sell and upsell revenue

💳 Domain: Finance & Banking

4. Credit Risk Assessment Model

- **Dataset**: Credit Card Approval or Loan Default

- **Goals**: Predict loan default probability, risk scoring
- **Key Features**: Feature engineering, class imbalance handling, model explainability
- **Business Value**: Reduce default rates and optimize lending decisions

5. Fraud Detection System

- **Dataset**: Credit Card Fraud or Online Payments Fraud
- **Goals**: Real-time fraud detection, anomaly identification
- **Key Features**: Anomaly detection, streaming data simulation, alert system
- **Business Value**: Prevent fraudulent transactions in real-time

▼ 6. Stock Market Analytics & Trading Signals

- **Dataset**: NYSE/NASDAQ Data or Yahoo Finance API
- **Goals**: Technical indicators, sentiment analysis, trading signals
- **Key Features**: Time series analysis, moving averages, volatility metrics
- **Business Value**: Data-driven investment decisions

▼ 🏥 Domain: Healthcare & Life Sciences

▼ 7. Patient Readmission Prediction

- **Dataset**: Hospital Readmissions or Diabetes Dataset
- **Goals**: Predict 30-day readmission risk, identify high-risk patients
- **Key Features**: Clinical feature engineering, survival analysis
- **Business Value**: Improve patient outcomes and reduce costs

▼ 8. Medical Diagnosis Assistant

- **Dataset**: Heart Disease or Cancer Detection
- **Goals**: Disease classification, risk factor analysis
- **Key Features**: Multi-class classification, feature importance, SHAP values

- **Business Value**: Support clinical decision-making

▼ 📱 Domain: Social Media & Content

▼ 9. Social Media Sentiment Analytics

- **Dataset**: Twitter Sentiment or Reddit Comments
- **Goals**: Real-time sentiment tracking, trend detection, influencer identification
- **Key Features**: NLP, topic modeling, sentiment time series
- **Business Value**: Brand monitoring and crisis detection

▼ 10. Content Recommendation System

- **Dataset**: Netflix Movies or Spotify Songs
- **Goals**: Personalized content recommendations
- **Key Features**: Collaborative filtering, content-based filtering, hybrid models
- **Business Value**: Increase user engagement and retention

▼ 🚗 Domain: Transportation & Logistics

▼ 11. Ride-Sharing Demand Prediction

- **Dataset**: NYC Taxi or Uber/Lyft Data
- **Goals**: Predict demand hotspots, optimize driver allocation
- **Key Features**: Geospatial analysis, time series forecasting
- **Business Value**: Reduce wait times and maximize driver utilization

▼ 12. Supply Chain Optimization

- **Dataset**: Supply Chain Data or Delivery Orders
- **Goals**: Inventory optimization, delivery route optimization
- **Key Features**: Network analysis, constraint optimization
- **Business Value**: Reduce costs and improve delivery times

▼ 🎮 Domain: Gaming & Entertainment

▼ 13. Player Behavior Analytics

- **Dataset**: Steam Games or Video Game Sales
- **Goals**: Churn prediction, engagement scoring, monetization optimization
- **Key Features**: Cohort analysis, player segmentation, LTV prediction
- **Business Value**: Increase player retention and revenue

▼ 14. Movie Revenue Prediction

- **Dataset**: TMDB Movies or IMDB Data
- **Goals**: Predict box office revenue, identify success factors
- **Key Features**: Feature engineering (cast, genre, budget), ensemble models
- **Business Value**: Investment decision support for studios

---

▼ 🌍 Domain: Sustainability & Environment

▼ 15. Energy Consumption Forecasting

- **Dataset**: Household Energy or Smart Grid
- **Goals**: Predict energy demand, optimize grid management
- **Key Features**: Time series forecasting, seasonality detection
- **Business Value**: Reduce waste and optimize energy distribution

▼ 16. Air Quality Prediction System

- **Dataset**: Air Quality UCI or Global Air Pollution
- **Goals**: Forecast pollution levels, identify contributing factors
- **Key Features**: Multi-variate time series, spatial analysis
- **Business Value**: Public health alerts and policy recommendations

---

▼ 🏢 Domain: HR & Workforce Analytics

▼ 17. Employee Attrition Prediction

- **Dataset**: IBM HR Analytics or HR Employee Data

- **Goals**: Predict employee turnover, identify retention strategies
- **Key Features**: Survival analysis, feature importance, segmentation
- **Business Value**: Reduce hiring costs and improve retention

18. Resume Screening & Talent Matching

- **Dataset**: Resume Dataset or Job Postings
- **Goals**: Auto-screen resumes, match candidates to roles
- **Key Features**: NLP, text classification, similarity matching
- **Business Value**: Accelerate hiring and improve candidate quality

---

🛠️ Technical Architecture Suggestions

Recommended Tech Stack

```
Data Ingestion Layer:
├── Kaggle API / External APIs
├── File uploads (CSV, JSON, Parquet)
└── Streaming simulation (optional)

Storage Layer (Delta Lake):
├── Bronze: Raw ingestion
├── Silver: Cleaned & validated
└── Gold: Business aggregates

Processing Layer:
├── PySpark DataFrames
├── Spark SQL
└── pandas UDFs (for complex logic)

Orchestration:
├── Databricks Workflows
├── Parameter widgets
└── Scheduled jobs

Analytics Layer:
├── Databricks SQL
├── Interactive dashboards
└── Visualization libraries (matplotlib, plotly)

ML Layer (if applicable):
├── Feature engineering
├── MLflow experiments
├── Model registry
└── Batch/real-time scoring

Governance:
├── Unity Catalog
```

```
├── Access controls
└── Data lineage
```

▼ 📋 Project Development Checklist (**Week Structure)**

▼ Days 15-16: Planning & Setup

- [ ] Define problem statement and success metrics
- [ ] Find and explore dataset (EDA)
- [ ] Design architecture (draw diagram)
- [ ] Set up Unity Catalog structure
- [ ] Create project repository/notebook structure

▼ Days 17-19: Implementation

- [ ] Build medallion pipeline (Bronze → Silver → Gold)
- [ ] Implement data quality checks
- [ ] Create business logic transformations
- [ ] Develop ML model (if applicable)
- [ ] Set up orchestration workflow
- [ ] Optimize performance (ZORDER, caching)

▼ Days 20-21: Polish & Documentation

- [ ] Build SQL dashboard or visualizations
- [ ] Create comprehensive documentation
- [ ] Write README with setup instructions
- [ ] Document key findings and insights
- [ ] Record demo video (optional but impressive!)
- [ ] Prepare presentation slides

▼ 💎 Tips for Success

▼ Do's:

✅ **Start Simple**: MVP first, then iterate
✅ **Think Business**: Every technical decision should have business justification
✅ **Document Early**: Write README as you build, not after
✅ **Use Git**: Version control your notebooks (Databricks Repos)
✅ **Test Incrementally**: Don't wait until the end to test
✅ **Optimize Smart**: Profile before optimizing (don't guess)
✅ **Tell a Story**: Your project should answer: What? Why? How? So What?

▼ Don't's:

❌ **Overscope**: Better to nail a smaller project than half-finish a huge one
❌ **Ignore Data Quality**: Garbage in = garbage out
❌ **Skip Documentation**: Future you (and employers) will thank you
❌ **Reinvent Wheels**: Use existing libraries and patterns
❌ **Forget Users**: Always think about who will use this

▼ 🎨 Making Your Project Stand Out

▼ Beyond the Basics:

1. **Interactive Elements**
   - Parameter widgets for dynamic queries
   - Real-time metric updates
   - Interactive dashboards with drill-downs
2. **Production Readiness**
   - Error handling and logging
   - Data validation checks
   - Monitoring and alerts
   - Automated testing
3. **Advanced ML**
   - Hyperparameter tuning with MLflow
   - Model explainability (SHAP, LIME)
   - A/B testing framework
   - Model monitoring for drift

4. **Business Impact**
    - ROI calculations
    - Before/after comparisons
    - Cost-benefit analysis
    - Actionable recommendations
5. **Presentation**
    - Executive summary (1-pager)
    - Architecture diagram
    - Demo video walkthrough
    - Key metrics dashboard

▼ 📊 Self Evaluation Framework

How to assess your own project:

Category	Questions to Ask	Weight
Problem Definition	Is the problem clear and valuable?	10%
Data Engineering	Is the pipeline robust and scalable?	25%
Technical Execution	Are best practices followed?	25%
Business Value	Does it solve a real problem?	20%
Documentation	Can others understand and reproduce?	10%
Innovation	Any creative or novel approaches?	10%

Target Score: 80%+ for portfolio-worthy projects

▼ 🔗 Recommended Data Sources

- **Kaggle Datasets**: kaggle.com/datasets
- **UCI ML Repository**: archive.ics.uci.edu
- **Google Dataset Search**: datasetsearch.research.google.com
- **AWS Open Data**: registry.opendata.aws
- **Data.gov**: data.gov

- **APIs**: Twitter, Reddit, Financial APIs, Weather APIs

▼ 🎓 Final Words

> 💡 Your capstone is more than just a project—it's proof that you can:
> ✨ **Think like a data engineer**: Design scalable architectures
> 📊 **Analyze like a data scientist**: Extract insights from data
> 💼 **Communicate like a consultant**: Present business value clearly
> **Remember**: The best projects are those that:
> Solve problems YOU care about
> Demonstrate skills YOU want to showcase
> Create value OTHERS can see
> Don't just complete a project—**build something you're proud of**. This is what employers want to see in your portfolio.

▼ 📝 Submission Guidelines (Optional)

When sharing your capstone:

▼ Required Artifacts:

1. **GitHub Repository** (or Databricks Repo)
   - Well-organized notebooks
   - README with setup instructions
   - Requirements/dependencies file
2. **Architecture Diagram**
   - Data flow visualization
   - Component relationships
3. **Documentation**
   - Problem statement
   - Methodology
   - Key findings
   - Business recommendations

4. **Demo**
    - Dashboard screenshots OR
    - Video walkthrough (5-10 min) OR
    - Live Databricks notebook

▼ Optional but Impressive:

- Blog post explaining your approach
- LinkedIn article about insights
- YouTube tutorial using your project
- Contribution to open-source dataset

🚀 You've Got This!

You've spent 14 days learning Databricks—now it's time to prove what you can do. Don't wait for the perfect idea. Pick something interesting, start building, and iterate. The journey from raw data to insights is where the magic happens.

Your challenge: Build something that makes you think, "I can't believe I made this in a week!"

Good luck, and happy building! 🎉