

Project: Job Market ETL Pipeline (MySQL)

Description: This script establishes the foundational data layer for the Job Market Analysis dashboard. It handles schema definition, data ingestion, and critical schema transformations (Wide-to-Long unpivoting) to prepare the dataset for BI consumption.

1. Database Initialization

Establishes a dedicated namespace for the project to ensure isolation from other system schemas.

SQL

```
-- Create a dedicated database for the job market project
CREATE DATABASE job_market;
```

```
-- Use the database
USE job_market;
```

2. Raw Ingestion Layer (Schema-on-Read)

Adopts an ELT (Extract, Load, Transform) strategy. All columns are initialized as TEXT to ensure 100% successful ingestion, deferring strict type casting to the transformation layer (Power BI/Power Query) to prevent data loss during the import process.

SQL

```
-- Create raw jobs table
-- All columns are kept as TEXT to avoid CSV import failures
-- Data cleaning and type conversion will be handled later in Power BI
```

```
CREATE TABLE jobs_raw (
  Title TEXT,
  Company TEXT,
  starRating TEXT,
  reviewsCount TEXT,
  experience TEXT,
  salary TEXT,
  location TEXT,
```

```
job_description TEXT,  
skill_1 TEXT,  
skill_2 TEXT,  
skill_3 TEXT,  
skill_4 TEXT,  
skill_5 TEXT,  
skill_6 TEXT,  
skill_7 TEXT,  
skill_8 TEXT,  
posted_on TEXT,  
Dept TEXT  
);
```

3. Ingestion Integrity Checks

Performs sanity checks immediately after loading to verify record counts and validate that the dataset captures the target domain (Data & Analytics roles).

SQL

```
-- Check total number of records imported  
SELECT COUNT(*) AS total_jobs  
FROM jobs_raw;  
  
-- Validate relevant analytical scope  
-- Counts jobs related to Data & Analyst roles  
SELECT COUNT(*) AS data_analyst_jobs  
FROM jobs_raw  
WHERE Title LIKE '%Data%'  
OR Title LIKE '%Analyst%';
```

4. Schema Transformation: Normalization (Unpivoting)

Solves the "Wide Table" problem. The raw data violates 1NF (First Normal Form) by storing repeating values (Skills) across columns. This step transforms the data from Wide format to Long format, establishing a proper One-to-Many relationship between Jobs and Skills.

SQL

```
-- Create a long-format skill table  
-- Each row represents one job-skill relationship
```

```
CREATE TABLE jobs_skills AS
```

```
SELECT Title, Dept, skill_1 AS skill  
FROM jobs_raw  
WHERE skill_1 IS NOT NULL
```

```
UNION ALL  
SELECT Title, Dept, skill_2  
FROM jobs_raw  
WHERE skill_2 IS NOT NULL
```

```
UNION ALL  
SELECT Title, Dept, skill_3  
FROM jobs_raw  
WHERE skill_3 IS NOT NULL
```

```
UNION ALL  
SELECT Title, Dept, skill_4  
FROM jobs_raw  
WHERE skill_4 IS NOT NULL
```

```
UNION ALL  
SELECT Title, Dept, skill_5  
FROM jobs_raw  
WHERE skill_5 IS NOT NULL
```

```
UNION ALL  
SELECT Title, Dept, skill_6  
FROM jobs_raw  
WHERE skill_6 IS NOT NULL
```

```
UNION ALL  
SELECT Title, Dept, skill_7  
FROM jobs_raw  
WHERE skill_7 IS NOT NULL
```

```
UNION ALL  
SELECT Title, Dept, skill_8  
FROM jobs_raw  
WHERE skill_8 IS NOT NULL;
```

5. Transformation Validation

Verifies the cardinality of the new dataset. The row count should increase significantly, confirming that the unpivoting logic correctly captured all skills across the columns.

SQL

```
-- Confirm row expansion after unpivot
-- Row count should be significantly higher than jobs_raw
SELECT COUNT(*) AS total_skill_rows
FROM jobs_skills;
```

```
-- Identify most in-demand skills
SELECT skill, COUNT(*) AS demand_count
FROM jobs_skills
GROUP BY skill
ORDER BY demand_count DESC
LIMIT 10;
```

6. Standardization & Grouping Logic

Demonstrates logic to categorize fragmented text data (e.g., MySQL, PostgreSQL, T-SQL) into standardized analytical categories. (Note: Full implementation deferred to Power Query for iterative flexibility).

SQL

```
-- Normalize SQL-related skills into a single category
-- This prevents fragmented skill counts (MYSQL, POSTGRESQL, etc.)
SELECT
  Title,
  Dept,
  CASE
    WHEN skill LIKE '%SQL%' THEN 'SQL'
    ELSE skill
  END AS sql_grouped
FROM jobs_skills;
```

7. Pipeline Egress (Data Export)

Exports the transformed and normalized datasets into clean flat files, ready for ingestion into the BI tool.

SQL

```
-- Export raw jobs data to CSV (server-side export)
```

```
SELECT *  
FROM jobs_raw  
INTO OUTFILE '/var/lib/mysql-files/jobs_raw.csv'  
FIELDS TERMINATED BY ','  
ENCLOSED BY '"'  
LINES TERMINATED BY '\n';
```

```
-- Export unpivoted skills data
```

```
SELECT *  
FROM jobs_skills  
INTO OUTFILE '/var/lib/mysql-files/jobs_skills.csv'  
FIELDS TERMINATED BY ','  
ENCLOSED BY '"'  
LINES TERMINATED BY '\n';
```

Technical Competencies Demonstrated

- **ETL Architecture:** Decoupling the data layer (SQL) from the reporting layer (Power BI).
- **Data Modeling:** converting non-normalized data (Wide) into analyzable facts (Long).
- **Data Quality:** Implementing integrity checks at the ingestion stage.
- **Granularity Control:** Understanding how to handle one-to-many relationships without causing fan-out errors in the final dashboard.