Compendio

🎯 Obiettivo

Creare un **report HR** che unisca dati di:

- dipendenti (`employees.csv`),
- stipendi (`salaries.csv`),
- performance (`performance.csv`).

Il report deve contenere:

- statistiche per reparto/ruolo,
- rilevamento outlier sugli stipendi,
- ranking dei top performer,
- esportazione in Excel e CSV.

🛠️ Step 0 — Setup ambiente

1. Crea la struttura:

```
hr-report/
  report_hr.py
  data/
  output/
```

2. Installa le librerie:

```
pip install pandas numpy openpyxl
```

3. Metti i CSV dentro `data/`.

📄 Step 1 — Caricamento dati (`load_data`)

- Usa `pd.read_csv()` per aprire i file.
- Per la colonna `hire_date` ricordati `parse_dates=["hire_date"]`.

- Se viene passato un anno (`performance_year`), filtra il DataFrame delle performance su quell'anno.

👉 **TODO**: implementa il caricamento e il filtro.

🔗 Step 2 — Merge dati (`merge_data`)

- Unisci employees + salaries su `employee_id` con un **left join**.
- Poi aggiungi performance con un altro **merge**.
- Gestisci i valori mancanti:
  - se `bonus` è NaN → sostituisci con 0.
- Aggiungi una nuova colonna: `total_comp = base_salary + bonus`.

👉 **TODO**: scrivi il codice dentro la funzione `merge_data()`.

🧹 Step 3 — Pulizia (`clean_data`)

- Rimuovi **duplicati** su `employee_id`.
- Assicurati che `hire_date` sia di tipo datetime.
- Droppa righe con NaN in:
  - `base_salary` (critico),
  - `rating` (per semplicità nell'esercizio).

👉 **TODO**: completa `clean_data()`.

📊 Step 4 — Aggregazioni (`aggregate_by_dept_role`)

- Raggruppa per `department` + `role`.
- Calcola:
  - numero dipendenti,
  - media, mediana e deviazione standard di `base_salary` e `total_comp`,
  - media di `rating`.
- Ordina per reparto e ruolo.

👉 **TODO**: implementa `aggregate_by_dept_role()` con `groupby().agg()`.

🚨 Step 5 — Outlier con IQR (`detect_outliers_iqr`)

- Per ogni `department`, calcola:
  - Q1 = 25° percentile,
  - Q3 = 75° percentile,
  - IQR = Q3 - Q1,
  - Limiti: `lower = Q1 - 1.5*IQR`, `upper = Q3 + 1.5*IQR`.
- Un dipendente è outlier se `total_comp < lower` o `> upper`.
- Aggiungi colonna booleana `is_comp_outlier`.

👉 **TODO**: completa `detect_outliers_iqr()` usando `groupby().transform()`.

🏆 Step 6 — Ranking (`build_rankings`)

- Normalizza i valori **per reparto**:
  - `rating_norm` = rating normalizzato (min-max),
  - `goals_norm` = goals normalizzato (min-max).
- Calcola lo score:

```
perf_score = 0.7*rating_norm + 0.3*goals_norm
```

- Crea:
  - **top_performers_by_dept** = top 10 per ogni reparto (ordinati per score decrescente),
  - **top_performers_global** = top 10 globali.

👉 **TODO**: implementa `build_rankings()`.

📤 Step 7 — Export (`export_report`)

- KPI minimi da calcolare:
  - numero dipendenti (`nunique`),
  - retribuzione totale media,
  - rating medio.

- Crea file Excel con più sheet (`ExcelWriter`):
  - KPI,
  - Aggregati,
  - TopByDept,
  - TopGlobal.
- Esporta CSV solo con gli outlier.

👉 **TODO**: completa `export_report()`.

🧪 Step 8 — Test

- Esegui:

```
python report_hr.py --test
```

- I test controlleranno:
  - Merge corretto (`total_comp` giusto per id 101 e 104).
  - Aggregazioni giuste per Engineering/Developer.
  - Colonna `is_comp_outlier` presente.
  - Ranking globale con id 104 primo.
  - Export: file Excel e CSV creati.

📝 Suggerimenti extra

- Usa `df.head()` spesso per controllare i dati intermedi.
- Con `df.info()` vedi i tipi di dato e i NaN.
- Se qualcosa non torna, stampa solo le colonne utili:

```
print(df[["employee_id","total_comp","rating"]].head())
```

- Procedi milestone per milestone: **scrivi codice → salva → lancia test**.
