1. /\*

2. Elasticsearch Lab Exercises

3.

4. https://georgebridgeman.com/exercises/olympic-data/olympics-01/

5.

6. Exercise 03

7. Download the dataset from here and use Kibana’s Data Visualizer to upload the file into a new index called olympic-events.

8. \*/

9.

10. /\*

11.

12. Exercise 04

13. Validate that the data was imported correctly by using a single API call to show the index name, index health, number of documents, and the size of the primary store. The details in the response must be in that order, with headers, and for the new index only.

14. \*/

15.

16. GET \_cat/indices/olympic-events?v

17. GET \_cat/indices/olympic-events?v&h=index,health,docs.count,pri.store.size

18.

19. /\*

20. Exercise 07

21. Look at how Elasticsearch has applied very general-purpose mappings to the data. Why has it chosen to use a keyword type for the Age field? Find all unique values for the Age field; there are less than 100 unique values for the Age field. Look for any suspicious values.

22. \*/

23.

24. GET olympic-events/\_mapping

25. GET olympic-events/\_search?filter\_path=aggregations

26. {

27. "size": 0,

28. "aggs": {

29. "unique\_ages": {

30. "terms": {

31. "field": "Age",

32. "size": 100

33. }

34. },

35. "number\_of\_unique\_ages": {

36. "cardinality": {

37. "field": "Age"

38. }

39. }

40. }

41. }

42.

43. /\*

44. Exercise 08

45. We will be deleting data in the next exercise; making a backup is always prudent. Without making any changes to the data, reindex the olympic-events index into a new index called olympic-events-backup.

46. \*/

47. PUT olympic-events-backup

48.

49. POST \_reindex

50. {

51. "source": {

52. "index": "olympic-events"

53. },

54. "dest": {

55. "index": "olympic-events-backup"

56. }

57. }

58.

59. /\*

60. Exercise 09

61. The Height and Weight fields suffer from the same problem as the Age field. Later exercises will require numeric-type queries for these fields so we want to exclude any document we can’t use in our analyses. In a single request, delete all documents from the olympic-events index that have a value of NA for either the Age, Height or Weight field.

62. \*/

63.

64. POST olympic-events/\_delete\_by\_query

65. {

66. "query": {

67. "multi\_match": {

68. "query": "NA",

69. "fields": [

70. "Age",

71. "Height",

72. "Weight"]

73. }

74. }

75. }

76.

77. /\*

78. Exercise 10

79. Notice how the Games field contains both the Olympic year and season. Create an ingest pipeline called split\_games that will split this field into two new fields - year and season - and remove the original Games field.

80. \*/

81.

82. GET olympic-events/\_search

83. {

84. "size": 0,

85. "aggs": {

86. "Game Names": {

87. "terms": {

88. "field": "Games",

89. "size": 10000

90. }

91. },

92. "Unique Game Counts": {

93. "cardinality": {

94. "field": "Games"

95. }

96. }

97. }

98. }

99.

100. PUT \_ingest/pipeline/split\_games

101. {

102. "processors": [

103. {"split": {

104. "field": "Games",

105. "separator": " "

106. }},

107. {

108. "script": {

109. "lang": "painless",

110. "source": """

111. ctx.year = ctx.Games[0];

112. ctx.season = ctx.Games[1];

113. """

114. }

115. },

116. {

117. "remove": {

118. "field": "Games"

119. }

120. }

121. ]

122. }

123.

124. /\*

125. Exercise 11

126. Ensure your new pipeline is working correctly by simulating it with these values:

127.

128. 1998 Summer

129. 2014 Winter

130. \*/

131.

132. POST \_ingest/pipeline/split\_games/\_simulate

133. {

134. "docs": [

135. {

136. "\_source": {

137. "Games": "1998 Summer"

138. }

139. }

140. ]

141. }

142.

143. /\*

144. Exercise 12

145. We’ll now start to clean up the mappings. Create a new index called olympic-events-fixed with 1 shard, 0 replicas, and the following mapping:

146.

147. Field Type

148. athleteId integer

149. age short

150. height short

151. weight short

152. athleteName text + keyword

153. gender keyword

154. team keyword

155. noc keyword

156. year short

157. season keyword

158. city text + keyword

159. sport keyword

160. event text + keyword

161. medal keyword

162. \*/

163.

164. GET olympic-events/\_mapping

165.

166. PUT olympic-events-fixed

167. {

168. "settings": {

169. "number\_of\_shards": 1,

170. "number\_of\_replicas": 0

171. },

172. "mappings": {

173. "properties": {

174. "athleteId": {

175. "type": "integer"

176. },

177. "age": {

178. "type": "short"

179. },

180. "height": {

181. "type": "short"

182. },

183. "weight": {

184. "type": "short"

185. },

186. "athleteName": {

187. "type": "text",

188. "fields": {

189. "keyword": {

190. "type": "keyword"

191. }

192. }

193. },

194. "gender": {

195. "type": "keyword"

196. },

197. "team": {

198. "type": "keyword"

199. },

200. "noc": {

201. "type": "keyword"

202. },

203. "year": {

204. "type": "short"

205. },

206. "season": {

207. "type": "keyword"

208. },

209. "city": {

210. "type": "text",

211. "fields": {

212. "keyword": {

213. "type": "keyword"

214. }

215. }

216. },

217. "sport": {

218. "type": "keyword"

219. },

220. "event": {

221. "type": "text",

222. "fields": {

223. "keyword": {

224. "type": "keyword"

225. }

226. }

227. },

228. "medal": {

229. "type": "keyword"

230. }

231. }

232. }

233. }

234.

235. /\*

236. Exercise 13

237. Reindex the data in the olympic-events index into the new olympic-events-fixed index created in exercise 12 using the split\_games pipeline created in exercise 10.

238. \*/

239.

240. POST \_reindex

241. {

242. "source": {

243. "index": "olympic-events"

244. },

245. "dest": {

246. "index": "olympic-events-fixed",

247. "pipeline": "split\_games"

248. }

249. }

250.

251. GET olympic-events-fixed/\_mapping

252.

253. /\*

254. Exercise 14

255. Look at the mapping for the olympic-events-fixed index. Notice how Elasticsearch has created new fields. We created the mapping for this index with the same field names as before but we put all the field names in lowercase. Field names are case sensitive, so Age and age are different, distinct fields to Elasticsearch.

256.

257. Also notice that the new mapping uses athleteId instead of ID, athleteName instead of Name and gender instead of Sex.

258.

259. We’ll need to correct this by tearing down the new index and reindexing with an additional pipeline to use the correct field names. To save us constantly having to recreate the index with the right mappings, we can leverage index templates.

260.

261. Create an index template called olympic-events for new indices with a name beginning with olympic-events-. Use the mapping and settings we defined in exercise 12 and configure the mapping so Elasticsearch will throw an exception if a document contains a field not defined in the mapping.

262. \*/

263.

264. PUT \_template/olympic-events

265. {

266. "index\_patterns": ["olympic-events-\*"],

267. "settings": {

268. "number\_of\_shards": 1,

269. "number\_of\_replicas": 0

270. },

271. "mappings": {

272. "dynamic": "strict",

273. "properties": {

274. "athleteId": {

275. "type": "integer"

276. },

277. "age": {

278. "type": "short"

279. },

280. "height": {

281. "type": "short"

282. },

283. "weight": {

284. "type": "short"

285. },

286. "athleteName": {

287. "type": "text",

288. "fields": {

289. "keyword": {

290. "type": "keyword"

291. }

292. }

293. },

294. "gender": {

295. "type": "keyword"

296. },

297. "team": {

298. "type": "keyword"

299. },

300. "noc": {

301. "type": "keyword"

302. },

303. "year": {

304. "type": "short"

305. },

306. "season": {

307. "type": "keyword"

308. },

309. "city": {

310. "type": "text",

311. "fields": {

312. "keyword": {

313. "type": "keyword"

314. }

315. }

316. },

317. "sport": {

318. "type": "keyword"

319. },

320. "event": {

321. "type": "text",

322. "fields": {

323. "keyword": {

324. "type": "keyword"

325. }

326. }

327. },

328. "medal": {

329. "type": "keyword"

330. }

331. }

332. }

333. }

334.

335. /\*

336. Exercise 15

337. Create a new ingest pipeline called reconcile\_fields to replace all fields with their correct field names (except for the Games field), then also execute the split\_games pipeline

338. \*/

339. PUT \_ingest/pipeline/reconcile\_fields

340. {

341. "processors": [

342. {

343. "rename": {

344. "field": "ID",

345. "target\_field": "athleteId"

346. }

347. },

348. {

349. "rename": {

350. "field": "Name",

351. "target\_field": "athleteName"

352. }

353. },

354. {

355. "rename": {

356. "field": "Age",

357. "target\_field": "age"

358. }

359. },

360. {

361. "rename": {

362. "field": "Height",

363. "target\_field": "height"

364. }

365. },

366. {

367. "rename": {

368. "field": "Weight",

369. "target\_field": "weight"

370. }

371. },

372. {

373. "rename": {

374. "field": "Sex",

375. "target\_field": "gender"

376. }

377. },

378. {

379. "rename": {

380. "field": "Team",

381. "target\_field": "team"

382. }

383. },

384. {

385. "rename": {

386. "field": "NOC",

387. "target\_field": "noc"

388. }

389. },

390. {

391. "rename": {

392. "field": "Sport",

393. "target\_field": "sport"

394. }

395. },

396. {

397. "rename": {

398. "field": "Event",

399. "target\_field": "event"

400. }

401. },

402. {

403. "rename": {

404. "field": "City",

405. "target\_field": "city"

406. }

407. },

408. {

409. "rename": {

410. "field": "Medal",

411. "target\_field": "medal"

412. }

413. },

414. {

415. "pipeline": {

416. "name": "split\_games"

417. }

418. }

419. ]

420. }

421.

422. /\*

423. Exercise 16

424. Test your new pipeline with the following document:

425. \*/

426.

427. POST \_ingest/pipeline/reconcile\_fields/\_simulate

428. {

429. "docs": [{

430. "\_source": {

431. "NOC": "ARG",

432. "Sex": "M",

433. "City": "Los Angeles",

434. "Weight": "98",

435. "Name": "Ernesto Arturo Alas",

436. "Sport": "Shooting",

437. "Games": "1984 Summer",

438. "Event": "Shooting Men's Free Pistol, 50 metres",

439. "Height": "186",

440. "Team": "Argentina",

441. "ID": 2224,

442. "Medal": "NA",

443. "Age": "54"

444. }}]

445. }

446.

447. /\*

448. Exercise 17

449. Delete the olympic-events-fixed index.

450. \*/

451.

452. DELETE olympic-events-fixed

453.

454. /\*

455. Exercise 18

456. Reindex the data in the olympic-events index into a new olympic-events-fixed index using the reconcile\_fields pipeline. If Elasticsearch throws any exceptions, you may have missed a field in your pipeline.

457. \*/

458.

459. GET olympic-events/\_mapping

460.

461. POST \_reindex

462. {

463. "source": {

464. "index": "olympic-events"

465. },

466. "dest": {

467. "index": "olympic-events-fixed",

468. "pipeline": "reconcile\_fields"

469. }

470. }

471.

472. GET olympic-events-fixed/\_mapping

473.

474. /\*

475. Exercise 19

476. Write a single query to find the name of all Gymnastics events. There are less than 100 Gymnastics event types.

477. \*/

478.

479. GET olympic-events-fixed/\_search?filter\_path=aggregations

480. {

481. "query": {

482. "term": {

483. "sport": {

484. "value": "Gymnastics"

485. }

486. }

487. },

488. "aggs": {

489. "<100\_Gymnastics\_event\_types": {

490. "terms": {

491. "field": "event.keyword",

492. "size": 100

493. }

494. },

495. "Gymnastics\_event\_count": {

496. "cardinality": {

497. "field": "event.keyword"

498. }

499. }

500. }

501. }

502.

503. /\*

504. Exercise 20

505. Write a single query to find the average weight for male and female competitors in Gymnastics events.

506. \*/

507. GET olympic-events-fixed/\_search?filter\_path=aggregations

508. {

509. "size": 0,

510. "query": {

511. "term": {

512. "sport": {

513. "value": "Gymnastics"

514. }

515. }

516. },

517. "aggs": {

518. "average\_weight\_male\_and\_female\_combined": {

519. "avg": {

520. "field": "weight"

521. }

522. }

523. }

524. }

525.

526. GET olympic-events-fixed/\_search?filter\_path=aggregations

527. {

528. "size": 0,

529. "query": {

530. "term": {

531. "sport": {

532. "value": "Gymnastics"

533. }

534. }

535. },

536. "aggs": {

537. "avg\_weight\_by\_genders": {

538. "terms": {

539. "field": "gender"

540. },

541. "aggs": {

542. "avg\_weight": {

543. "avg": {

544. "field": "weight"

545. }

546. }

547. }

548. }

549. }

550. }

551.

552. /\*

553. Exercise 21

554. Write a single query to find the year that each of the 590 unique events first appeared in the Olympic Games, and which events were introduced most recently.

555. \*/

556.

557. GET olympic-events-fixed/\_search?filter\_path=aggregations

558. {

559. "size": 0,

560. "aggs": {

561. "event\_year\_where\_it\_was\_first\_appeared": {

562. "terms": {

563. "field": "event.keyword",

564. "size": 590

565. },

566. "aggs": {

567. "first\_year\_of\_apparence": {

568. "min": {

569. "field": "year"

570. }

571. }

572. }

573. }

574. }

575. }

576.

577. /\*

578. Write a query to return only the following fields for the 50 tallest athletes in the 2016 Rio de Janeiro Games:

579.

580. athleteName

581. team

582. sport

583. age

584. height

585. weight

586. gender

587. \*/

588.

589. GET olympic-events-fixed/\_search

590. {

591. "\_source": ["athleteName.keyword", "city","team", "sport", "age", "height", "weight", "gender"],

592. "query": {

593. "bool": {

594. "must": [

595. {"match": {

596. "year": "2016"

597. }},

598. {"match": {

599. "city.keyword": "Rio de Janeiro"

600. }}

601. ]

602. }

603. },

604. "aggs": {

605. "50\_tallest\_athletes": {

606. "terms": {

607. "size": 50,

608. "field": "height",

609. "order": {

610. "\_key": "desc"

611. }

612. }

613. }

614. }

615. }

616.

617. /\*

618. Exercise 23

619. The weight and height fields are in metric. Weight is in kg and height is in cm. Add a scripted field called weightLbs to the previous query to return the weight in lbs. The formula for this is: Weight \* 2.2

620. \*/

621. PUT olympic-events-fixed/\_mapping

622. {

623. "runtime": {

624. "weightLbs": {

625. "type": "double",

626. "script": {

627. "source": "emit(doc['weight'].value \* 2.2)"

628. }

629. }

630. }

631. }

632.

633. GET olympic-events-fixed/\_mapping

634.

635. GET olympic-events-fixed/\_search?filter\_path=aggregations

636. {

637. "aggs": {

638. "athletes' weight in Lbs": {

639. "terms": {

640. "field": "weightLbs",

641. "size": 1000,

642. "order": {

643. "\_key": "desc"

644. }

645. }

646. }

647. }

648. }

649.

650. /\*

651. Exercise 24

652. Add a scripted field called bmi to the previous query to return the BMI for each athlete, calculated using the following formula: Weight / (Height in m squared)

653. \*/

654.

655. PUT olympic-events-fixed/\_mapping

656. {

657. "runtime": {

658. "bmi": {

659. "type": "double",

660. "script": {

661. "source": "emit(doc['weight'].value / (Math.pow(doc['height'].value / 100.0f, 2)))"

662. }

663. }

664. }

665. }

666.

667. GET olympic-events-fixed/\_mapping

668.

669. GET olympic-events-fixed/\_search?filter\_path=aggregations

670. {

671. "aggs": {

672. "athletes' bmi": {

673. "terms": {

674. "field": "bmi",

675. "size": 1000,

676. "order": {

677. "\_key": "desc"

678. }

679. }

680. }

681. }

682. }

683.

684. /\*

685. Exercise 25

686. Write a query to return the first 50 documents for gold medal athletics events, in descending age order.

687. \*/

688. GET olympic-events-fixed/\_search

689. {

690. "size": 50,

691. "query": {

692. "bool": {

693. "must": [

694. {"match": {

695. "sport": "Athletics"

696. }},

697. {"match": {

698. "medal": "Gold"

699. }}

700. ]

701. }

702. },

703. "sort": [

704. {

705. "age": {

706. "order": "desc"

707. }

708. }

709. ]

710. }

711.

712. /\*

713. Exercise 26

714. Write a query to match swimming events where either:

715.

716. The athlete’s weight was between 60kg and 70kg

717. The athlete’s age was less than 20

718. Enhance the query so the results identify whether the weight, age, or both matched the search criteria.

719. \*/

720. GET olympic-events-fixed/\_search

721. {

722. "size": 10000,

723. "query": {

724. "bool": {

725. "should": [

726. {"range": {

727. "weight": {

728. "gt": 60,

729. "lt": 70

730. }

731. }},

732. {"range": {

733. "age": {

734. "lt": 20

735. }

736. }}

737. ],

738. "minimum\_should\_match": 1,

739. "must": [

740. {"match": {

741. "sport": "Swimming"

742. }}

743. ]

744. }

745. }

746. }

747.

748. /\*

749. Exercise 28

750. Change the number of replica shards for the olympic-noc-regions index to be 0. The index should then be green and contain 230 documents.

751. \*/

752. PUT olympic-noc-regions

753. {

754. "settings": {

755. "number\_of\_replicas": 0

756. }

757. }

758.

759. GET \_cat/indices/olympic-noc-regions?v

760.