

BI / read / 1

BI 1

BI 2

BI 3

BI 4

BI 5

BI 6

BI 7

BI 8

BI 9

BI 10

BI 11

BI 12

BI 13

BI 14

BI 15

BI 16

BI 17

BI 18

BI 19

BI 20

query	BI / read / 1																																							
title	Posting summary																																							
pattern	<table><tr><td colspan="2">message: Message</td></tr><tr><td colspan="2">creationDate < \$dateTime</td></tr><tr><td>length</td><td>year(creationDate)</td></tr></table>					message: Message		creationDate < \$dateTime		length	year(creationDate)																													
message: Message																																								
creationDate < \$dateTime																																								
length	year(creationDate)																																							
desc.	<p>Given a datetime, find all Messages created before that moment. Group them by a 3-level grouping:</p> <ol style="list-style-type: none">by year of creationfor each year, group into Message types: is Comment or notfor each year-type group, split into four groups based on length of their content<ul style="list-style-type: none">0: 0 ≤ length < 40 (short)1: 40 ≤ length < 80 (one liner)2: 80 ≤ length < 160 (tweet)3: 160 ≤ length (long)																																							
params	<table><tr><td>1</td><td>datetime</td><td>DateTime</td><td colspan="2">For later microbatches, later datetime parameters are selected keep the variance low (<0.5%)</td></tr></table>	1	datetime	DateTime	For later microbatches, later datetime parameters are selected keep the variance low (<0.5%)																																			
1	datetime	DateTime	For later microbatches, later datetime parameters are selected keep the variance low (<0.5%)																																					
result	<table><tr><td>1</td><td>year</td><td>32-bit Integer</td><td>R</td><td>year(message.creationDate)</td></tr><tr><td>2</td><td>isComment</td><td>Boolean</td><td>M</td><td>True for Comments, False for Posts</td></tr><tr><td>3</td><td>lengthCategory</td><td>32-bit Integer</td><td>C</td><td>0 for short, 1 for one-liner, 2 for tweet, 3 for long</td></tr><tr><td>4</td><td>messageCount</td><td>32-bit Integer</td><td>A</td><td>Total number of Messages in that group</td></tr><tr><td>5</td><td>averageMessageLength</td><td>32-bit Float</td><td>A</td><td>Average length of the Message content in that group</td></tr><tr><td>6</td><td>sumMessageLength</td><td>32-bit Integer</td><td>A</td><td>Sum of all Message content lengths</td></tr><tr><td>7</td><td>percentageOfMessages</td><td>32-bit Float</td><td>A</td><td>Number of Messages in group as a percentage of all messages created before the given date</td></tr></table>					1	year	32-bit Integer	R	year(message.creationDate)	2	isComment	Boolean	M	True for Comments, False for Posts	3	lengthCategory	32-bit Integer	C	0 for short, 1 for one-liner, 2 for tweet, 3 for long	4	messageCount	32-bit Integer	A	Total number of Messages in that group	5	averageMessageLength	32-bit Float	A	Average length of the Message content in that group	6	sumMessageLength	32-bit Integer	A	Sum of all Message content lengths	7	percentageOfMessages	32-bit Float	A	Number of Messages in group as a percentage of all messages created before the given date
1	year	32-bit Integer	R	year(message.creationDate)																																				
2	isComment	Boolean	M	True for Comments, False for Posts																																				
3	lengthCategory	32-bit Integer	C	0 for short, 1 for one-liner, 2 for tweet, 3 for long																																				
4	messageCount	32-bit Integer	A	Total number of Messages in that group																																				
5	averageMessageLength	32-bit Float	A	Average length of the Message content in that group																																				
6	sumMessageLength	32-bit Integer	A	Sum of all Message content lengths																																				
7	percentageOfMessages	32-bit Float	A	Number of Messages in group as a percentage of all messages created before the given date																																				
sort	<table><tr><td>1</td><td>year</td><td>↓</td><td colspan="2"></td></tr><tr><td>2</td><td>isComment</td><td>↑</td><td colspan="2">False < True, i.e. Posts come first and Comments second</td></tr><tr><td>3</td><td>lengthCategory</td><td>↑</td><td colspan="2">order based on the lengthCategory value</td></tr></table>					1	year	↓			2	isComment	↑	False < True, i.e. Posts come first and Comments second		3	lengthCategory	↑	order based on the lengthCategory value																					
1	year	↓																																						
2	isComment	↑	False < True, i.e. Posts come first and Comments second																																					
3	lengthCategory	↑	order based on the lengthCategory value																																					
limit	n/a																																							
CPs	1.2, 3.2, 4.1, 4.2, 8.5																																							

BI / read / 2

query	BI / read / 2																							
title	Tag evolution																							
pattern																								
desc.	Find the Tags under a given TagClass that were used in Messages during in the 100-day period starting at date and compare it with the 100-day period that follows. For the Tags and for both months, compute the count of Messages.																							
params	<table><tr><td>1</td><td>date</td><td>Date</td><td></td></tr><tr><td>2</td><td>tagClass</td><td>Long String</td><td>TagClasses with a similar amount of Messages are selected</td></tr></table>	1	date	Date		2	tagClass	Long String	TagClasses with a similar amount of Messages are selected															
1	date	Date																						
2	tagClass	Long String	TagClasses with a similar amount of Messages are selected																					
result	<table><tr><td>1</td><td>tag.name</td><td>Long String</td><td>R</td><td></td></tr><tr><td>2</td><td>countWindow1</td><td>32-bit Integer</td><td>A</td><td>Occurrences of the tagClass during the first time window</td></tr><tr><td>3</td><td>countWindow2</td><td>32-bit Integer</td><td>A</td><td>Occurrences of the tagClass during the second time window</td></tr><tr><td>4</td><td>diff</td><td>32-bit Integer</td><td>A</td><td>Absolute difference of countWindow1 and countWindow2</td></tr></table>	1	tag.name	Long String	R		2	countWindow1	32-bit Integer	A	Occurrences of the tagClass during the first time window	3	countWindow2	32-bit Integer	A	Occurrences of the tagClass during the second time window	4	diff	32-bit Integer	A	Absolute difference of countWindow1 and countWindow2			
1	tag.name	Long String	R																					
2	countWindow1	32-bit Integer	A	Occurrences of the tagClass during the first time window																				
3	countWindow2	32-bit Integer	A	Occurrences of the tagClass during the second time window																				
4	diff	32-bit Integer	A	Absolute difference of countWindow1 and countWindow2																				
sort	<table><tr><td>1</td><td>diff</td><td>↓</td><td></td></tr><tr><td>2</td><td>tag.name</td><td>↑</td><td></td></tr></table>	1	diff	↓		2	tag.name	↑																
1	diff	↓																						
2	tag.name	↑																						
limit	100																							
CPs	2.4, 3.1, 3.2, 4.1, 4.2, 4.3, 5.3, 6.1, 8.2, 8.5																							

BI / read / 3

query	BI / read / 3				
title	Popular topics in a country				
pattern	<pre>classDiagram class Country { name = \$country } class City { } class Person { id } class Forum { id title creationDate } class TagClass { name = \$tagClass } class Tag { } class Message { message } class Post { } Country < -- City City --> Forum : isLocatedIn Person --> Forum : hasModerator Forum --> Post : containerOf TagClass --> Tag : hasType Tag --> Message : hasTag Message --> Post : replyOf*0..</pre>				
desc.	<p>Given a TagClass and a Country, find all the Forums created in the given Country, containing at least one Message with Tags belonging directly to the given TagClass, and count the Messages by the Person who created it and by the Forum which contains them.</p> <p>The location of a Forum is identified by the location of the Forum’s moderator.</p>				
params	1	tagClass	Long String	TagClasses with a similar amount of Messages are selected	
	2	country	Long String	Big Countries are selected	
result	1	forum.id	ID	R	
	2	forum.title	Long String	R	
	3	forum.creationDate	DateTime	R	
	4	person.id	ID	R	
	5	messageCount	32-bit Integer	A	
sort	1	messageCount	↓		
	2	forum.id	↑		
limit	20				
CPs	1.1, 1.2, 1.3, 2.1, 2.2, 2.4, 3.3, 8.2				

BI / read / 4

query	BI / read / 4				
title	Top message creators by country				
pattern	<div><div><div>1. select top 100 forums based on memberCount in country</div><div><div>Country</div><div>name</div></div><div>isPartOf</div><div><div>City</div><div>isLocatedIn</div><div>memberCount = count(member)</div><div><div>member: Person</div><div>hasMember</div><div><div>forum: Forum</div><div>creationDate > \$date</div></div></div></div></div><div><div>2. for each country, for each of the top 100 forums (topForum), find the top posters</div><div><div><div>topForum: Forum</div><div>containerOf</div><div><div>Post</div><div>replyOf*0..</div><div><div>messageCount = count(message)</div><div><div>Message</div><div>creationDate > \$date</div></div></div></div><div><div>Forum</div><div>is in top 100 forum</div><div>hasMember</div><div><div>person: Person</div><div>id</div><div>firstName</div><div>lastName</div><div>creationDate</div></div></div><div><div>hasCreator</div></div></div></div></div></div>				
desc.	<p>Find the most popular Forums by Country, where the popularity of a Forum is measured by the number of members that Forum has from a given Country.</p> <p>Calculate the top 100 most popular Forums. If a Forum is popular in multiple countries, it should only be calculated once with its largest membership. In case of a tie, the Forum(s) with the smaller id value(s) should be selected.</p> <p>For each member Person of the 100 most popular Forums, count the number of Messages (messageCount) they made in any of those (most popular) Forums. Also include those member Persons who have not posted any Messages (have a messageCount of 0).</p>				
params	1	date	Date	Selected from the first 30 days of the network	
result	1	person.id	ID	R	
	2	person.firstName	String	R	
	3	person.lastName	String	R	
	4	person.creationDate	DateTime	R	
	5	messageCount	32-bit Integer	A	
sort	1	messageCount	↓		
	2	person.id	↑		
limit	100				
CPs	1.2, 1.3, 2.1, 2.2, 2.3, 2.4, 3.3, 5.3, 6.1, 8.2, 8.4				

BI / read / 5

query	BI / read / 5			
title	Most active posters of a given topic			
pattern				
desc.	<p>Get each Person (person) who has created a Message (message) with a given Tag (direct relation, not transitive). Considering only these Messages, for each Person node:</p> <ul style="list-style-type: none"> Count its messages (messageCount). Count likes (likeCount) to its messages. Count Comments (replyCount) in reply to it messages. <p>The score is calculated according to the following formula: $1 \times \text{messageCount} + 2 \times \text{replyCount} + 10 \times \text{likeCount}$.</p>			
params	1	tag	Long String	Tags with a similar amount of Messages are selected
result	1	person.id	ID	R
	2	replyCount	32-bit Integer	A
	3	likeCount	32-bit Integer	A
	4	messageCount	32-bit Integer	A
	5	score	32-bit Integer	A
sort	1	score	↓	
	2	person.id	↑	
limit	100			
CPs	1.2, 2.3, 8.2			

BI / read / 6

query	BI / read / 6			
title	Most authoritative users on a given topic			
pattern	<pre> graph TD Tag[Tag] -- hasTag --> message1[message1: Message] person1[person: Person] -- hasCreator --> message1 message1 -- "«opt» likes" --> p2[p2: Person] subgraph Compute_p2_popularityScore [Compute p2.popularityScore] p2 -- hasCreator --> message2[message2: Message] message2 -- "«opt» likes" --> p3[p3: Person] p2 -- "p2.popularityScore = count(p3)" --> p3 end p2 -- "person.authorityScore = sum(p2.popularityScore)" --> p2 </pre>			
desc.	<p>Given a Tag (tag), find all Persons (person) that ever created a Message with the Tag. For each of these Persons (person) compute their “authority score” as follows:</p> <ul style="list-style-type: none"> The “authority score” is the sum of “popularity scores” of the Persons (p2) that liked any of that Person’s Messages with the given Tag (same criterion as for message1). A Person’s (p2) “popularity score” is defined as the total number of likes on all of their Messages (message2). 			
params	1	tag	Long String	Tags with a similar amount of Messages are selected
result	1	person.id	ID	R
	2	authorityScore	32-bit Integer	A
sort	1	authorityScore	↓	
	2	person1.id	↑	
limit	100			
CPs	1.2, 2.3, 3.3, 6.1, 8.2			
relevance	Computing the authority scores might involve computing the popularity score for the same Person multiple times. Implementations are advised to avoid such redundant computations.			

BI / read / 7

BI 1	query	BI / read / 7			
BI 2	title	Related topics			
BI 3	pattern				
BI 4	BI 5	BI 6	BI 7	BI 8	BI 9
BI 10	desc.	Find all Messages that have a given Tag. Find the related Tags attached to (direct) reply Comments of these Messages, but only of those reply Comments that do not have the given Tag. Group the Tags by name, and get the count of replies in each group.			
BI 11	params	1	tag	Long String	Tags with a similar amount of Messages are selected
BI 12	BI 13	BI 14	BI 15	BI 16	BI 17
BI 18	BI 19	BI 20	BI 21	BI 22	BI 23
BI 24	BI 25	BI 26	BI 27	BI 28	BI 29
BI 30	BI 31	BI 32	BI 33	BI 34	BI 35
BI 36	BI 37	BI 38	BI 39	BI 40	BI 41
BI 42	BI 43	BI 44	BI 45	BI 46	BI 47
BI 48	BI 49	BI 50	BI 51	BI 52	BI 53
BI 54	BI 55	BI 56	BI 57	BI 58	BI 59
BI 60	BI 61	BI 62	BI 63	BI 64	BI 65
BI 66	BI 67	BI 68	BI 69	BI 70	BI 71
BI 72	BI 73	BI 74	BI 75	BI 76	BI 77
BI 78	BI 79	BI 80	BI 81	BI 82	BI 83
BI 84	BI 85	BI 86	BI 87	BI 88	BI 89
BI 90	BI 91	BI 92	BI 93	BI 94	BI 95
BI 96	BI 97	BI 98	BI 99	BI 100	BI 101
BI 102	BI 103	BI 104	BI 105	BI 106	BI 107
BI 108	BI 109	BI 110	BI 111	BI 112	BI 113
BI 114	BI 115	BI 116	BI 117	BI 118	BI 119
BI 120	BI 121	BI 122	BI 123	BI 124	BI 125
BI 126	BI 127	BI 128	BI 129	BI 130	BI 131
BI 132	BI 133	BI 134	BI 135	BI 136	BI 137
BI 138	BI 139	BI 140	BI 141	BI 142	BI 143
BI 144	BI 145	BI 146	BI 147	BI 148	BI 149
BI 150	BI 151	BI 152	BI 153	BI 154	BI 155
BI 156	BI 157	BI 158	BI 159	BI 160	BI 161
BI 162	BI 163	BI 164	BI 165	BI 166	BI 167
BI 168	BI 169	BI 170	BI 171	BI 172	BI 173
BI 174	BI 175	BI 176	BI 177	BI 178	BI 179
BI 180	BI 181	BI 182	BI 183	BI 184	BI 185
BI 186	BI 187	BI 188	BI 189	BI 190	BI 191
BI 192	BI 193	BI 194	BI 195	BI 196	BI 197
BI 198	BI 199	BI 200	BI 201	BI 202	BI 203
BI 204	BI 205	BI 206	BI 207	BI 208	BI 209
BI 210	BI 211	BI 212	BI 213	BI 214	BI 215
BI 216	BI 217	BI 218	BI 219	BI 220	BI 221
BI 222	BI 223	BI 224	BI 225	BI 226	BI 227
BI 228	BI 229	BI 230	BI 231	BI 232	BI 233
BI 234	BI 235	BI 236	BI 237	BI 238	BI 239
BI 240	BI 241	BI 242	BI 243	BI 244	BI 245
BI 246	BI 247	BI 248	BI 249	BI 250	BI 251
BI 252	BI 253	BI 254	BI 255	BI 256	BI 257
BI 258	BI 259	BI 260	BI 261	BI 262	BI 263
BI 264	BI 265	BI 266	BI 267	BI 268	BI 269
BI 270	BI 271	BI 272	BI 273	BI 274	BI 275
BI 276	BI 277	BI 278	BI 279	BI 280	BI 281
BI 282	BI 283	BI 284	BI 285	BI 286	BI 287
BI 288	BI 289	BI 290	BI 291	BI 292	BI 293
BI 294	BI 295	BI 296	BI 297	BI 298	BI 299
BI 300	BI 301	BI 302	BI 303	BI 304	BI 305
BI 306	BI 307	BI 308	BI 309	BI 310	BI 311
BI 312	BI 313	BI 314	BI 315	BI 316	BI 317
BI 318	BI 319	BI 320	BI 321	BI 322	BI 323
BI 324	BI 325	BI 326	BI 327	BI 328	BI 329
BI 330	BI 331	BI 332	BI 333	BI 334	BI 335
BI 336	BI 337	BI 338	BI 339	BI 340	BI 341
BI 342	BI 343	BI 344	BI 345	BI 346	BI 347
BI 348	BI 349	BI 350	BI 351	BI 352	BI 353
BI 354	BI 355	BI 356	BI 357	BI 358	BI 359
BI 360	BI 361	BI 362	BI 363	BI 364	BI 365
BI 366	BI 367	BI 368	BI 369	BI 370	BI 371
BI 372	BI 373	BI 374	BI 375	BI 376	BI 377
BI 378	BI 379	BI 380	BI 381	BI 382	BI 383
BI 384	BI 385	BI 386	BI 387	BI 388	BI 389
BI 390	BI 391	BI 392	BI 393	BI 394	BI 395
BI 396	BI 397	BI 398	BI 399	BI 400	BI 401
BI 402	BI 403	BI 404	BI 405	BI 406	BI 407
BI 408	BI 409	BI 410	BI 411	BI 412	BI 413
BI 414	BI 415	BI 416	BI 417	BI 418	BI 419
BI 420	BI 421	BI 422	BI 423	BI 424	BI 425
BI 426	BI 427	BI 428	BI 429	BI 430	BI 431
BI 432	BI 433	BI 434	BI 435	BI 436	BI 437
BI 438	BI 439	BI 440	BI 441	BI 442	BI 443
BI 444	BI 445	BI 446	BI 447	BI 448	BI 449
BI 450	BI 451	BI 452	BI 453	BI 454	BI 455
BI 456	BI 457	BI 458	BI 459	BI 460	BI 461
BI 462	BI 463	BI 464	BI 465	BI 466	BI 467
BI 468	BI 469	BI 470	BI 471	BI 472	BI 473
BI 474	BI 475	BI 476	BI 477	BI 478	BI 479
BI 480	BI 481	BI 482	BI 483	BI 484	BI 485
BI 486	BI 487	BI 488	BI 489	BI 490	BI 491
BI 492	BI 493	BI 494	BI 495	BI 496	BI 497
BI 498	BI 499	BI 500	BI 501	BI 502	BI 503
BI 504	BI 505	BI 506	BI 507	BI 508	BI 509
BI 510	BI 511	BI 512	BI 513	BI 514	BI 515
BI 516	BI 517	BI 518	BI 519	BI 520	BI 521
BI 522	BI 523	BI 524	BI 525	BI 526	BI 527
BI 528	BI 529	BI 530	BI 531	BI 532	BI 533
BI 534	BI 535	BI 536	BI 537	BI 538	BI 539
BI 540	BI 541	BI 542	BI 543	BI 544	BI 545
BI 546	BI 547	BI 548	BI 549	BI 550	BI 551
BI 552	BI 553	BI 554	BI 555	BI 556	BI 557
BI 558	BI 559	BI 560	BI 561	BI 562	BI 563
BI 564	BI 565	BI 566	BI 567	BI 568	BI 569
BI 570	BI 571	BI 572	BI 573	BI 574	BI 575
BI 576	BI 577	BI 578	BI 579	BI 580	BI 581
BI 582	BI 583	BI 584	BI 585	BI 586	BI 587
BI 588	BI 589	BI 590	BI 591	BI 592	BI 593
BI 594	BI 595	BI 596	BI 597	BI 598	BI 599
BI 600	BI 601	BI 602	BI 603	BI 604	BI 605
BI 606	BI 607	BI 608	BI 609	BI 610	BI 611
BI 612	BI 613	BI 614	BI 615	BI 616	BI 617
BI 618	BI 619	BI 620	BI 621	BI 622	BI 623
BI 624	BI 625	BI 626	BI 627	BI 628	BI 629
BI 630	BI 631	BI 632	BI 633	BI 634	BI 635
BI 636	BI 637	BI 638	BI 639	BI 640	BI 641
BI 642	BI 643	BI 644	BI 645	BI 646	BI 647
BI 648	BI 649	BI 650	BI 651	BI 652	BI 653
BI 654	BI 655	BI 656	BI 657	BI 658	BI 659
BI 660	BI 661	BI 662	BI 663	BI 664	BI 665
BI 666	BI 667	BI 668	BI 669	BI 670	BI 671
BI 672	BI 673	BI 674	BI 675	BI 676	BI 677
BI 678	BI 679	BI 680	BI 681	BI 682	BI 683
BI 684	BI 685	BI 686	BI 687	BI 688	BI 689
BI 690	BI 691	BI 692	BI 693	BI 694	BI 695
BI 696	BI 697	BI 698	BI 699	BI 700	BI 701
BI 702	BI 703	BI 704	BI 705	BI 706	BI 707
BI 708	BI 709	BI 710	BI 711	BI 712	BI 713
BI 714	BI 715	BI 716	BI 717	BI 718	BI 719
BI 720	BI 721	BI 722	BI 723	BI 724	BI 725
BI 726	BI 727	BI 728	BI 729	BI 730	BI 731
BI 732	BI 733	BI 734	BI 735	BI 736	BI 737
BI 738	BI 739	BI 740	BI 741	BI 742	BI 743
BI 744	BI 745	BI 746	BI 747	BI 748	BI 749
BI 750	BI 751	BI 752	BI 753	BI 754	BI 755
BI 756	BI 757	BI 758	BI 759	BI 760	BI 761
BI 762	BI 763	BI 764	BI 765	BI 766	BI 767
BI 768	BI 769	BI 770	BI 771	BI 772	BI 773
BI 774	BI 775	BI 776	BI 777	BI 778	BI 779
BI 780	BI 781	BI 782	BI 783	BI 784	BI 785
BI 786	BI 787	BI 788	BI 789	BI 790	BI 791
BI 792	BI 793	BI 794	BI 795	BI 796	BI 797
BI 798	BI 799	BI 800	BI 801	BI 802	BI 803
BI 804	BI 805	BI 806	BI 807	BI 808	BI 809
BI 810	BI 811	BI 812	BI 813	BI 814	BI 815
BI 816	BI 817	BI 818	BI 819	BI 820	BI 821
BI 822	BI 823	BI 824	BI 825	BI 826	BI 827
BI 828	BI 829	BI 830	BI 831	BI 832	BI 833
BI 834	BI 835	BI 836	BI 837	BI 838	BI 839
BI 840	BI 841	BI 842	BI 843	BI 844	BI 845
BI 846	BI 847	BI 848	BI 849	BI 850	BI 851
BI 852	BI 853	BI 854	BI 855	BI 856	BI 857
BI 858	BI 859	BI 860	BI 861	BI 862	BI 863
BI 864	BI 865	BI 866	BI 867	BI 868	BI 869
BI 870	BI 871	BI 872	BI 873	BI 874	BI 875
BI 876	BI 877	BI 878	BI 879	BI 880	BI 881
BI 882	BI 883	BI 884	BI 885	BI 886	BI 887
BI 888	BI 889	BI 890	BI 891	BI 892	BI 893
BI 894	BI 895	BI 896	BI 897	BI 898	BI 899
BI 900	BI 901	BI 902	BI 903	BI 904	BI 905
BI 906	BI 907	BI 908	BI 909	BI 910	BI 911
BI 912	BI 913	BI 914	BI 915	BI 916	BI 917
BI 918	BI 919	BI 920	BI 921	BI 922	BI 923
BI 924	BI 925	BI 926	BI 927	BI 928	BI 929
BI 930	BI 931	BI 932	BI 933	BI 934	BI 935
BI 936	BI 937	BI 938	BI 939	BI 940	BI 941
BI 942	BI 943	BI 944	BI 945	BI 946	BI 947
BI 948	BI 949	BI 950	BI 951	BI 952	BI 953
BI 954	BI 955	BI 956	BI 957	BI 958	BI 959
BI 960	BI 961	BI 962	BI 963	BI 964	BI 965
BI 966	BI 967	BI 968	BI 969	BI 970	BI 971
BI 972	BI 973	BI 974	BI 975	BI 976	BI 977
BI 978	BI 979	BI 980	BI 981	BI 982	BI 983
BI 984	BI 985	BI 986	BI 987	BI 988	BI 989
BI 990	BI 991	BI 992	BI 993	BI 994	BI 995
BI 996	BI 997	BI 998	BI 999	BI 1000	BI 1001
BI 1002	BI 1003	BI 1004	BI 1005	BI 1006	BI 1007
BI 1008	BI 1009	BI 1010	BI 1011	BI 1012	BI 1013
BI 1014	BI 1015	BI 1016	BI 1017	BI 1018	BI 1019
BI 1020	BI 1021	BI 1022	BI 1023	BI 1024	BI 1025
BI 1026	BI 1027	BI 1028	BI 1029	BI 1030	BI 1031
BI 1032	BI 1033	BI 1034	BI 1035	BI 1036	BI 1037
BI 1038	BI 1039	BI 1040	BI 1041	BI 1042	BI 1043
BI 1044	BI 1045	BI 1046	BI 1047	BI 1048	BI 1049
BI 1050	BI 1051	BI 1052	BI 1053	BI 1054	BI 1055
BI 1056	BI 1057	BI 1058	BI 1059	BI 1060	BI 1061
BI 1062	BI 1063	BI 1064	BI 1065	BI 1066	BI 1067
BI 1068	BI 1069	BI 1070	BI 1071	BI 1072	BI 1073
BI 1074	BI 1075	BI 1076	BI 1077	BI 1078	BI 1079
BI 1080	BI 1081	BI 1082	BI 1083	BI	

BI / read / 8

query	BI / read / 8				
title	Central person for a tag				
pattern	<div><p>For each person with a matching hasInterest and/or hasCreator edge, compute person.score = ((if hasInterest edge exists then 100 else 0) + count(message))</p><p>Calculate the sum of the friends' scores: friendsScore = sum(friend.score)</p></div>				
desc.	<p>Given a Tag, find all Persons that are interested in the Tag and/or have written a Message (Post or Comment) with a creationDate after a given date and that has a given Tag. For each Person, compute the score as the sum of the following two aspects:</p> <ul style="list-style-type: none">• 100, if the Person has this Tag as their interest, or 0 otherwise• number of Messages by this Person with the given Tag <p>Also, for each Person, compute the sum of the score of the Person’s friends (friendsScore).</p>				
params	<div><div>1</div><div>tag</div></div>	<div><div>Long String</div></div>	<div>Tags with a similar amount of Messages are selected</div>		
	<div><div>2</div><div>date</div></div>	<div><div>Date</div></div>	<div>Dates from around the same day are selected. (TODO - how exactly? what distribution?)</div>		
result	<div><div>1</div><div>person.id</div></div>	<div><div>ID</div></div>	<div><div>R</div></div>		
	<div><div>2</div><div>score</div></div>	<div><div>32-bit Integer</div></div>	<div><div>A</div></div>		
	<div><div>3</div><div>friendsScore</div></div>	<div><div>32-bit Integer</div></div>	<div><div>A</div></div>	<div>The sum of the score of the person’s friends</div>	
sort	<div><div>1</div><div>score + friendsScore</div></div>	<div><div>↓</div></div>			
	<div><div>2</div><div>person.id</div></div>	<div><div>↑</div></div>			
limit	100				
CPs	1.2, 2.1, 2.3, 3.2, 5.3, 8.2, 8.4, 8.5				
relevance	Similarly to BI 16, there are two major ways to compute this query: (1) creating an induced subgraph of the interested Persons and their friends and performing the scoring on this graph or (2) performing the scoring without creating an induced subgraph and scoring the friends of a Person on-the-fly. The first approach is more efficient as it avoids redundant computations, however, specifying it needs support for composable graph queries.				

BI / read / 9

query	BI / read / 9				
title	Top thread initiators				
pattern					
desc.	<p>For each Person, count the number of Posts they created in the time interval [startDate, endDate] (equivalent to the number of threads they initiated) and the number of Messages in each of their (transitive) reply trees, including the root Post of each tree. When calculating Message counts only consider Messages created within the given time interval.</p> <p>Return each Person, number of Posts they created, and the count of all Messages that appeared in the reply trees (including the Post at the root of tree) they created.</p>				
params	1	startDate	Date	TODO	
	2	endDate	Date	8-10 days after the startDate	
result	1	person.id	ID	R	
	2	person.firstName	String	R	
	3	person.lastName	String	R	
	4	threadCount	32-bit Integer	A	The number of Posts created by that Person (the number of threads initiated)
	5	messageCount	32-bit Integer	A	The number of Messages created in all the threads this Person initiated
sort	1	messageCount	↓		
	2	person.id	↑		
limit	100				
CPs	1.2, 2.2, 2.3, 3.2, 7.2, 7.3, 7.4, 8.1, 8.5				

BI / read / 10

query	BI / read / 10				
title	Experts in social circle				
pattern					
desc.	<p>Given a Person (startPerson), find all other Persons (expertCandidatePerson) that live in a given Country and are connected to given Person by a <i>shortest path</i> with length in range [minPathDistance, maxPathDistance] through the knows relation.</p> <p>For each of these expertCandidatePerson nodes, retrieve all of their Messages that contain at least one Tag belonging to a given TagClass (direct relation not transitive). For each Message, retrieve all of its Tags.</p> <p>Group the results by Persons and Tags, then count the Messages by a certain Person having a certain Tag.</p>				
params	1	personId	ID	The ID of the startPerson. Persons with a similar degree of knows edges are selected	
	2	country	String	Countries with a similar number of Persons are selected	
	3	tagClass	Long String	TagClasses with a similar degree of hasType edges are selected	
	4	minPathDistance	32-bit Integer	1 or 2	
	5	maxPathDistance	32-bit Integer	2 or 3	
result	1	expertCandidatePerson.id	ID	R	
	2	tag.name	Long String	R	
	3	messageCount	32-bit Integer	A	Number of Messages created by that Person containing that Tag
sort	1	messageCount	↓		
	2	tag.name	↑		
	3	expertCandidatePerson.id	↑		
limit	100				
CPs	1.2, 1.3, 2.3, 2.4, 3.3, 5.3, 7.1, 7.2, 7.3, 8.1, 8.6				

BI / read / 11

query	BI / read / 11											
title	Friend triangles											
pattern	<pre>graph TD Country[Country] City1[City] City2[City] City3[City] PersonA[a: Person] PersonB[b: Person] PersonC[c: Person] Country -- isPartOf --> City1 Country -- isPartOf --> City2 Country -- isPartOf --> City3 City1 -- isLocatedIn --> PersonA City2 -- isLocatedIn --> PersonB City3 -- isLocatedIn --> PersonC PersonA -- knows --> PersonB PersonB -- knows --> PersonC PersonA -- knows --> PersonC</pre>											
desc.	<p>For a given country, count all the distinct triples of Persons such that:</p> <ul style="list-style-type: none">• a is friend of b,• b is friend of c,• c is friend of a, <p>and these friendships were created after a given startDate.</p> <p>Distinct means that given a triple t_1 in the result set R of all qualified triples, there is no triple t_2 in R such that t_1 and t_2 have the same set of elements.</p>											
params	<table><tr><td>1</td><td>country</td><td>Long String</td><td></td></tr><tr><td>2</td><td>startDate</td><td>Date</td><td></td></tr></table>	1	country	Long String		2	startDate	Date				
1	country	Long String										
2	startDate	Date										
result	<table><tr><td>1</td><td>count</td><td>32-bit Integer</td><td>A</td><td></td></tr></table>	1	count	32-bit Integer	A							
1	count	32-bit Integer	A									
limit	n/a											
CPs	1.1, 2.3, 2.5											

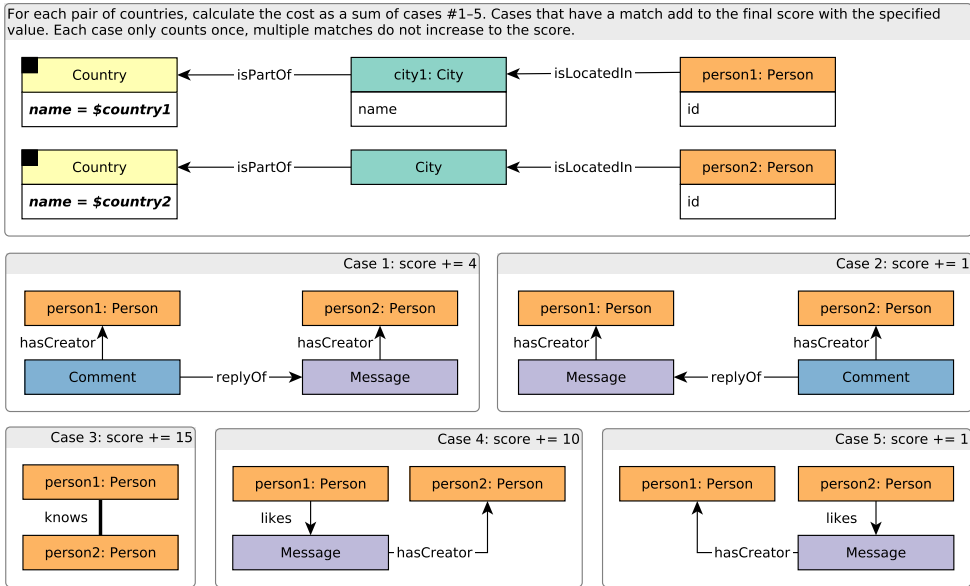
BI / read / 12

query	BI / read / 12				
title	How many persons have a given number of messages				
pattern	<div><div><div>2. personCount = count</div><div><div>Person</div></div><div>count Persons grouped by messageCount value</div></div><div><div>1. messageCount = count</div><div><div>Message</div><div>content not empty and length < \$lengthThreshold and \$date < \$creationDate</div></div></div><div><div>Post</div><div>language in \$languages</div></div><div><div>hasCreator</div><div>replyOf*0..</div></div><div><div>«opt»</div></div></div>				
desc.	<p>For each Person, count the number of Messages they made (messageCount). Only count Messages with the following attributes:</p> <ul style="list-style-type: none">• Its content is not empty (and consequently, the imageFile attribute is empty for Posts).• Its length is below the lengthThreshold (exclusive, equality is not allowed).• Its creationDate is after date (exclusive, equality is not allowed).• It is written in any of the given languages. <ul style="list-style-type: none">– The language of a Post is defined by its language attribute.– The language of a Comment is that of the Post that initiates the thread where the Comment replies to. <p>The Post and Comments in the reply tree’s path (from the Message to the Post) do not have to satisfy the constraints for content, length and creationDate.</p> <p>For each messageCount value, count the number of Persons with exactly messageCount Messages (with the required attributes).</p>				
params	<div><div>1</div><div>date</div></div>	<div><div>Date</div></div>	<div></div>		
	<div><div>2</div><div>lengthThreshold</div></div>	<div><div>32-bit Integer</div></div>	<div>Selected as balanced against date to filter around 30% of the Messages within a language and keep the variance low</div>		
	<div><div>3</div><div>languages</div></div>	<div><div>{String}</div></div>	<div>Only the most frequently used languages are selected</div>		
result	<div><div>1</div><div>messageCount</div></div>	<div><div>32-bit Integer</div></div>	<div><div>A</div></div>	<div>Number of Messages created</div>	
	<div><div>2</div><div>personCount</div></div>	<div><div>32-bit Integer</div></div>	<div><div>A</div></div>	<div>Number of Persons with messageCount Messages</div>	
sort	<div><div>1</div><div>personCount</div></div>	<div><div>↓</div></div>	<div></div>		
	<div><div>2</div><div>messageCount</div></div>	<div><div>↓</div></div>	<div></div>		
limit	n/a				
CPs	1.1, 1.2, 1.4, 3.2, 4.2, 4.3, 8.1, 8.2, 8.3, 8.4, 8.5				

BI / read / 13

query	BI / read / 13				
title	Zombies in a country				
pattern	<div><div>1. zombies = collect(zombie)</div><div><div><div>Country</div><div><i>name = \$country</i></div></div><div><div>City</div><div></div></div><div><div>zombie: Person</div><div><i>creationDate < \$endDate</i> <i>and (messageCount / months) < 1</i></div></div><div><div>message: Message</div><div><i>creationDate < \$endDate</i></div></div><div>isPartOf</div><div>isLocatedIn</div><div>hasCreator</div></div><div>messageCount = count(message)</div></div> <div><div>2. For each zombie IN zombies, calculate: zombieScore = zombieLikeCount / totalLikeCount</div><div><div><div>likerPerson: Person</div><div><i>creationDate < \$endDate</i></div></div><div><div>zombie: Person</div><div></div></div><div><div>Message</div><div></div></div><div><div>likerZombie: Person</div><div><i>creationDate < \$endDate</i> <i>and likerZombie IN zombies</i></div></div><div>totalLikeCount = count(likerPerson)</div><div>zombieLikeCount = count(likerZombie)</div><div>likes</div><div>hasCreator</div></div></div>				
desc.	<p>Find zombies within the given country, and return their zombie scores. A zombie is a Person created before the given endDate, which has created an average of [0, 1) Messages per month, during the time range between profile’s creationDate and the given endDate. The number of months spans the time range from the creationDate of the profile to the endDate with partial months on both end counting as one month (e.g. a creationDate of Jan 31 and an endDate of Mar 1 result in 3 months).</p> <p>For each zombie, calculate the following:</p> <ul style="list-style-type: none">zombieLikeCount: the number of likes received from other zombies.totalLikeCount: the total number of likes received.zombieScore: zombieLikeCount / totalLikeCount. If the value of totalLikeCount is 0, the zombieScore of the zombie should be 0.0. <p>For both zombieLikeCount and totalLikeCount, only consider likes received from profiles that were created before the given endDate.</p>				
params	1	country	Long String	Only the largest Countries are selected	
	2	endDate	Date	Selected from the last days of the initial data set	
result	1	zombie.id	ID	R	
	2	zombieLikeCount	32-bit Integer	A	
	3	totalLikeCount	32-bit Integer	A	
	4	zombieScore	64-bit Float	A	Determined as zombieLikeCount / totalLikeCount
sort	1	zombieScore	↓		
	2	zombie.id	↑		
limit	100				
CPs	1.2, 2.1, 2.3, 2.4, 3.2, 3.3, 4.2, 5.1, 5.3, 8.2, 8.4, 8.5				

BI / read / 14

BI 1	query	BI / read / 14			
BI 2	title	International dialog			
BI 3	pattern	<p>For each pair of countries, calculate the cost as a sum of cases #1-5. Cases that have a match add to the final score with the specified value. Each case only counts once, multiple matches do not increase to the score.</p> 			
BI 4					
BI 5					
BI 6					
BI 7					
BI 8					
BI 9					
BI 10					
BI 11					
BI 12					
BI 13					
BI 14	desc.	<p>Consider all pairs of people (person1, person2) such that one is located in a City of Country country1 and the other is located in a City of Country country2. For each City of Country country1, return the highest scoring pair. The score of a pair is defined as the sum of the subscores awarded for the following kinds of interaction. The initial value is score = 0.</p> <ol style="list-style-type: none"> 1. person1 has created a reply Comment to at least one Message by person2: score += 4 2. person1 has created at least one Message that person2 has created a reply to: score += 1 3. person1 and person2 know each other: score += 15 4. person1 liked at least one Message by person2: score += 10 5. person1 has created at least one Message that was liked by person2: score += 1 <p>Consequently, the maximum score a pair can obtain is: 4 + 1 + 15 + 10 + 1 = 31.</p> <p>This query has two variants based on whether the input parameters are selected as correlated (close countries) or uncorrelated (far countries).</p>			
BI 15					
BI 16					
BI 17					
BI 18					
BI 19					
BI 20					
	params	1	country1	Long String	<p>A: correlated with parameter country2, i.e. the countries are close and there are many Persons visiting both Countries.</p> <p>B: uncorrelated with parameter country2, i.e. the countries are afar and there are few Persons visiting both Countries.</p>
		2	country2	Long String	
	result	1	person1.id	ID	R
		2	person2.id	ID	R
		3	city1.name	Long String	R
		4	score	32-bit Integer	C
	sort	1	score	↓	
		2	person1.id	↑	
		3	person2.id	↑	
	limit	n/a			
	CPs	1.3, 1.4, 2.1, 3.1, 3.3, 5.1, 5.2, 5.3, 8.3, 8.4			

BI / read / 15

BI 1	query	BI / read / 15			
BI 2	title	Trusted connection paths through forums created in a given timeframe			
BI 3	pattern				
BI 4	desc.	<p>Given two Persons, find all (unweighted) shortest paths between these two Persons, in the subgraph induced by the knows relationship.</p> <p>Then, for each path calculate a weight. The nodes in the path are Persons, and the weight of a path is the sum of weights between every pair of consecutive Person nodes in the path.</p> <p>The weight for a pair of Persons is calculated based on their interactions:</p> <ul style="list-style-type: none"> • Every direct reply (by one of the Persons) to a Post (by the other Person) contributes 1.0. • Every direct reply (by one of the Persons) to a Comment (by the other Person) contributes 0.5. <p>Only consider Messages that were created in a Forum that was created within the timeframe (interval) [startDate, endDate]. Note that for Comments, the containing Forum is that of the Post that the comment (transitively) replies to. Also note that interactions are counted both ways.</p> <p>Return all paths with the Person IDs ordered by their weights descending.</p>			
BI 5	params	1	person1Id	ID	
BI 6		2	person2Id	ID	
BI 7		3	startDate	Date	
BI 8		4	endDate	Date	
BI 9	result	1	person.id	[ID]	C
BI 10		2	weight	64-bit Float	C
BI 11	sort	1	weight	↓	The order of paths with the same weight is unspecified
BI 12		2	personIds	↑	The IDs in the paths are used for lexicographical sorting
BI 13	limit	n/a			
BI 14	CPs	1.2, 2.1, 2.2, 2.4, 3.3, 5.1, 5.3, 7.2, 7.3, 7.5, 7.7, 8.1, 8.2, 8.3, 8.4, 8.5, 8.6			

BI / read / 16

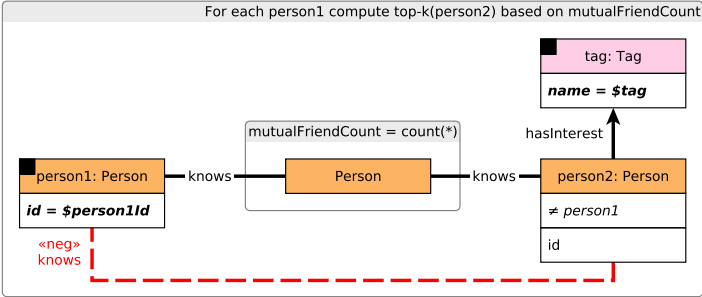
query	BI / read / 16				
title	Fake news detection				
pattern	<div><div>For \$tagX/\$dayX in [tagA/dateA, tagB/dateB], compute scoreX = count(messageX)</div><div><div>1. Create an induced subgraph of Persons who created a Message with Tag \$tagX on \$dateX</div><div><div><div>tag: Tag</div><div>name = \$tagX</div></div><div>hasTag</div><div><div>Message</div><div>day(creationDate) = \$dateX</div></div><div>hasCreator</div><div><div>person: Person</div></div></div></div><div><div>2. In the subgraph, count the Messages (using the same conditions) from People with \leq \$maxKnowsLimit friends</div><div><div><div>tag: Tag</div><div>name = \$tagX</div></div><div>hasTag</div><div><div>count(messageX)</div><div><div>messageX: Message</div><div>day(creationDate) = \$dateX</div></div></div><div>hasCreator</div><div><div>person: Person</div><div>count \leq \$maxKnowsLimit</div><div>«opt» knows</div><div>Person</div></div></div></div></div>				
desc.	<p>Given two Tag/date pairs (tagA/dateA and tagB/dateB), for each pair tagX/dateX:</p> <ul style="list-style-type: none">• Create an induced subgraph between Persons where for each pair of Persons person1/person2, both have created a Message on the day of dateX with Tag tagX.• In the induced subgraph, only keep pairs of Persons who have at most maxKnowsLimit friends (in the induced subgraph).• For these Persons, count the number of Messages created on dateX with Tag tagX. <p>Return Persons who had at least one Messages for both tagA/dateA and tagB/dateB ranked by their total number of Messages (descending).</p>				
params	1	tagA	Long String		
	2	dateA	Date		
	3	tagB	Long String		
	4	dateB	Date		
	5	maxKnowsLimit	32-bit Integer	Selected between 3 and 6	
result	1	person.id	ID	R	
	2	messageCountA	32-bit Integer	A	Message count for tagA/dateA
	3	messageCountB	32-bit Integer	A	Message count for tagB/dateB
sort	1	messageCountA + messageCountB	↓		
	2	person.id	↑		
limit	20				
CPs	5.3, 8.4, 8.5				
relevance	There are two major ways to compute this query: (1) create the induced subgraph as suggested by the specification (either as a view or in materialized form), or (2) skip creating the induced subgraph and perform on-the-fly check for the number of friends (who also posted at least one Message with the given Tag on the given date). The latter approach is easier to express in systems which do not provide graph views but might result in redundant computations (the query engine will might repeatedly check whether a Person has at least one Message that satisfies the conditions).				

BI / read / 17

query	BI / read / 17			
title	Information propagation analysis			
pattern				
desc.	<p>This query aims to identify instances of “information propagation” when a Person (person1) submits a Message (message1) with a given Tag (tag) to a Forum (forum1). This is read by other members of forum1, Persons person2 and person3. Some time later (specified by the delta parameter), these persons have a discussion with the same tag in a different Forum (forum2) where person1 is not a member. The discussion consists of a Message (message2) by person2 and a direct reply Comment (comment) by person3.</p> <p>Return IDs of person1 with the number of interactions their Messages (might have) caused.</p>			
params	1	tag	Long String	Tags with a similar amount of Messages are selected
	2	delta	32-bit Integer	Measured in hours, selected to be between 8 and 16 hours.
result	1	person1.id	ID	R
	2	messageCount	32-bit Integer	A
sort	1	person1.id	↑	
limit	n/a			
CPs	2.1, 2.3, 8.1			

BI / read / 18

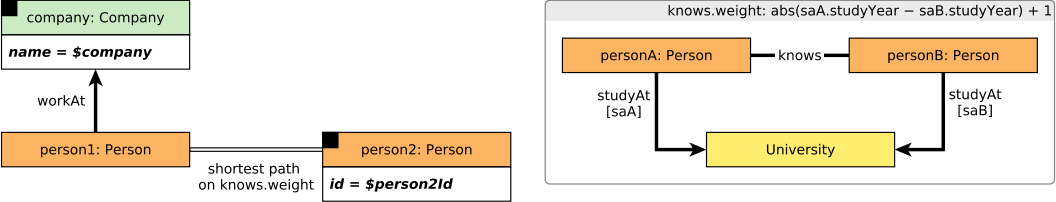
BI 1
BI 2
BI 3
BI 4
BI 5
BI 6
BI 7
BI 8
BI 9
BI 10
BI 11
BI 12
BI 13
BI 14
BI 15
BI 16
BI 17
BI 18
BI 19
BI 20

query	BI / read / 18			
title	Friend recommendation			
pattern	<div>For each person1 compute top-k(person2) based on mutualFriendCount</div>  <p>The diagram illustrates the query pattern for friend recommendation. It features three main entities: person1 (Person), a central Person, and person2 (Person). person1 is connected to the central Person via a 'knows' relationship. The central Person is connected to person2 via a 'knows' relationship. A calculation box labeled 'mutualFriendCount = count(*)' is positioned between the two 'knows' relationships. person2 is also connected to a Tag entity via a 'hasInterest' relationship. The Tag entity has a property 'name = \$tag'. A red dashed line with the label '«neg» knows' connects person1 to person2, indicating a negative relationship. The diagram is enclosed in a box with the title 'For each person1 compute top-k(person2) based on mutualFriendCount'.</p>			
desc.	<p>For a given Person (person1) and a Tag (tag), recommend new friends (person2) who</p> <ul style="list-style-type: none">• do not yet know person1• have many mutual friends with person1• are interested in tag. <p>Rank Persons person2 based on the number of mutual friends.</p>			
params	1	person1Id	ID	Persons with a similar amount of friends are selected
	2	tag	Long String	Tags with a similar amount of Messages are selected
result	1	person2.id	ID	R
	2	mutualFriendCount	32-bit Integer	A
sort	1	mutualFriendCount	↓	
	2	person2.id	↑	
limit	20			
CPs	2.5, 8.1			

BI / read / 19

BI 1	query	BI / read / 19			
BI 2	title	Interaction path between cities			
BI 3	pattern	<p>Find the shortest paths between all pairs of Persons in city1 and city2</p> <p>city1: City id = \$city1id</p> <p>city2: City id = \$city2id</p> <p>person1: Person person2: Person</p> <p>isLocatedIn compute weighted shortest paths on knows.weight</p> <p>The weight of a knows edge is based on the number of interactions between its Persons: knows.weight = 1 / (count(i1) + count(i2))</p> <p>Case i1: Reply from personA to Person B's Message</p> <p>Case i2: Reply from personB to personA's Message</p>			
BI 14	desc.	<p>Given two Cities <i>city1</i>, <i>city2</i>, find Persons <i>person1</i>, <i>person2</i> living in these Cities (respectively) with the shortest <i>interaction path</i> between them. If there are multiple pairs of people with shortest paths having the same total weight, return all of them.</p> <p>The shortest path is computed using a weight between two Persons defined as the reciprocal of the number of interactions (direct reply Comments to a Message by the other Person). Therefore, more interactions imply a smaller weight.</p> <p><i>Note:</i> Interactions are counted both ways, i.e. if Alice writes 2 reply Comments to Bob's Messages and Bob writes 3 reply Comments to Alice's Messages, their total number of interactions is 5.</p>			
BI 15	params	1	city1id	ID	Small Cities within the same Country are selected
BI 16		2	city2id	ID	
BI 17	result	1	person1.id	ID	R
BI 18		2	person2.id	ID	R
BI 19		3	totalWeight	64-bit Float	C
BI 20	sort	1	totalWeight	↑	
		2	person1.id	↑	
		3	person2.id	↑	
	limit	20			
	CPs	3.3, 7.6, 7.7, 8.4, 8.6			
	relevance	<p>Finding shortest paths between pairs of Persons in Cities can be implemented in theory with an <i>all-pairs shortest paths</i> algorithm. However, this needs to be executed on the whole Person-knows-Person graph (with edge weights derived from the number of interactions) so it is expected to be prohibitively expensive. A better approach is using multiple <i>single-source shortest path algorithms</i> (e.g. from the City with fewer inhabitants).</p>			

BI / read / 20

BI 1	query	BI / read / 20			
BI 2	title	Recruitment			
BI 3	pattern				
BI 10	desc.	<p>Given a Company company and a Person person2 (who is known to be working at another Company), find a Person (person1) working the in the company who have the top-20 shortest path to person2 through people who have studied together. On this path, we only consider edges between Persons who know each other and attended the same university and set the weight of the edge to the absolute difference between the year of enrolment plus 1 (<code>studyAt.classYear + 1</code>).</p> <p>If there are multiple Person person1 nodes with the same shortest path, return all of them.</p>			
BI 16	params	1	company	Long String	Companies with a similar number of employees (former or current) are selected
BI 17		2	person2Id	ID	
BI 18	result	1	person1.id	ID	R
BI 19		2	totalWeight	64-bit Integer	C
BI 20	sort	1	person1.id	↑	
	limit	20			
	CPs	3.3, 7.6, 7.7, 8.4, 8.6			
	relevance	Implementations can either pre-compute edge weights or compute them on-the-fly. To find a weighted shortest path efficiently, implementations can use e.g. a bidirectional Dijkstra algorithm.			