Num.	Story	Query	Explanation	Dataset
1	Some country VS the world	BEGIN CHILDREN {A1, A2} A1 LIKE [G,.*] and CHILDREN <B1, B2> B1 LIKE [F,country,eq,(?<Country>.*)] B2 LIKE [F,country,ne,(?<Country>.*)] A2 LIKE [G,.*] and CHILDREN <C1, C2> C1 LIKE [F,country,eq,(?<Country>.*)] C2 LIKE [F,country,ne,(?<Country>.*)]	This query is composed of two identical sub-queries. Each one of them is composed of group-by operation with some parameters, followed by two parallel filter operations: 1) filter the column country to some country 2) exclude the **same** country.	NETFLIX
2	Season2+	BEGIN CHILDREN <A1> A1 LIKE [F,type,eq,TV Show] and CHILDREN <B1> B1 LIKE [F,duration,ge,2] and CHILDREN <C1, C2> C1 LIKE [G,(?<col1>.*),.*] C2 LIKE [F,(?<col1>.*),.*] and CHILDREN <D1, D2> D1 LIKE [G,(?<col2>.*),.*] D2 LIKE [F,(?<col2>.*),.*]	First, filter the type to 'TV Show' and then filter to movies with at least 2 seasons. Next, apply some group-by operation and then apply some filter operation on the same aggregation key. The last step focuses the data to some subset that the previous group-by has raised. Then, repeat the last two steps on the current state in order to continue drilling down.	NETFLIX
3	Compare 3 actors in term of the same quality	A1 LIKE [F,cast,contains,.*] and SIBLINGS {A2,A3} and CHILDREN {B1} B1 LIKE [G,(?<col>.*),(?<func>.*),(?<agg>.*)] A2 LIKE [F,cast,contains,.*] and CHILDREN {B2} B2 LIKE [G,(?<col>.*),(?<func>.*),(?<agg>.*)] A3 LIKE [F,cast,contains,.*] and CHILDREN {B3} B3 LIKE [G,(?<col>.*),(?<func>.*),(?<agg>.*)]	For three times, filter to tuples that their cast column has some value, each time with different value. Then, for each one apply a group-by operation on the same column, employing the same aggregation on the same aggregated column.	NETFLIX
4	Average duration of movies of different subsets	BEGIN CHILDREN {A1} A1 LIKE [F,type,eq,Movie] and CHILDREN <B1, B2> B1 LIKE [F,.*] and DESCENDANTS {D1,*} D1 LIKE [G,.*,AVG,duration] B2 LIKE [F,.*] and DESCENDANTS {D2,*} D2 LIKE [G,.*,AVG,duration]	Filter the type to Movie. Then do twice: filter to some subset of the data and then group-by some column and aggregate according to the average movies duration.	NETFLIX
5	Investigate interesting properties of summer months flights in comparison to all flights	BEGIN CHILDREN <A1,A2,A3> A1 LIKE [G,(?<col1>.*),.*] A2 LIKE [G,(?<col2>.*),.*] A3 LIKE [F,Month,ge,7] and CHILDREN <B1> B1 LIKE [F,Month,le,8] and CHILDREN <C3,C4,*> C3 LIKE [G,(?<col1>.*),.*] C4 LIKE [G,(?<col2>.*),.*]	This query applies two group-by operations, and also another two group-by operations on the same columns as previously but specifically on flights between July and August.	FLIGHTS
6	Investigate various delay reasons	BEGIN CHILDREN <A1,A2,A3,A4> A1 LIKE [G,DelayReason,.*] A2 LIKE [F,DelayReason,eq,.*] and CHILDREN {B2} B2 LIKE [G,.*] A3 LIKE [F,DelayReason,eq,.*] and CHILDREN {B3} B3 LIKE [G,.*] A4 LIKE [F,DelayReason,eq,.*] and CHILDREN {B4} B4 LIKE [G,.*]	Group by delay reason, employing some aggregation on some column. Then for three times, focus on some delay reason and apply some group-by operation.	FLIGHTS
7	Find an interesting property of flights with long delays and drill down into the property	BEGIN CHILDREN {A1} A1 LIKE [F,DepartureDelay,eq,LARGE_DEALY] and CHILDREN <B1,*> B1 LIKE [G,(?<col>.*),.*] and SIBLINGS {B2, B3, B4} B2 LIKE [F,(?<col>.*),eq,.*] B3 LIKE [F,(?<col>.*),eq,.*] B4 LIKE [F,(?<col>.*),eq,.*]	Filter to flights that had a large departure delay. Then, group-by some column and show three subsets of the data by applying three filters on the column of the group-by operation.	FLIGHTS
8	At some point show interesting properties of flights with weather delay	BEGIN DESCENDANTS {D1,*} D1 LIKE [F,DelayReason,eq,WEATHER] and CHILDREN {B1, B2} B1 LIKE [G,.*] B2 LIKE [G,.*]	At some point filter to flights that delayed due to weather conditions, and then apply separately some two group-by operations.	FLIGHTS
9	How to get one million installs	BEGIN CHILDREN <A1,A2> A1 LIKE [G,(?<col>.*),(?<func>.*),(?<agg>.*)] and CHILDREN {B1} B1 LIKE [G,installs,.*] and CHILDREN {} A2 LIKE [F,installs,ge,1000000] and DESCENDANTS {D1,*} D1 LIKE [G,(?<col>.*),(?<func>.*),(?<agg>.*)]	Apply some group-by operation and then show some aggregation for each number of installs (bucketed). Also, apply the same first group-by operation on apps with at least one million installs.	PLAY STORE
10	Insights on apps with extremely high or low rating	BEGIN CHILDREN <A1,A2> A1 LIKE [F,rating,le,2.5] and CHILDREN <C1,C2> C1 LIKE [G,(?<col1>.*),(?<func1>.*),(?<agg1>.*)] C2 LIKE [G,(?<col2>.*),(?<func2>.*),(?<agg2>.*)] A2 LIKE [F,rating,ge,4.7] and CHILDREN <D1,D2,*> D1 LIKE [G,(?<col1>.*),(?<func1>.*),(?<agg1>.*)] D2 LIKE [G,(?<col2>.*),(?<func2>.*),(?<agg2>.*)]	Apply the same two group-by operations for: 1) apps with at most 2.5 rating. 2) apps with at least 4.7 rating.	PLAY STORE
11	Backwards compatibility	BEGIN CHILDREN <A1,A2,A3,A4> A1 LIKE [G,.*] and CHILDREN <B1> B1 LIKE [G,min_android_ver,CNT,app_id] and CHILDREN {} A2 LIKE [G,.*] and CHILDREN <C1,*> C1 LIKE [G,min_android_ver,.*] A3 LIKE [F,(?<col>.*),.*] and CHILDREN <D1,*> D1 LIKE [G,min_android_ver,.*] A4 LIKE [F,(?<col>.*),.*] and CHILDREN <E1,*> E1 LIKE [G,min_android_ver,.*]	This expression composed of 4 parts: 1. Apply some aggregation and then show the number of applications for each android version. 2. Apply some aggregation and then apply some other aggregation of each android version. 3. Filter some column to some value. 4. Filter the same column from previous step to some value.	PLAY STORE
12	Paid Vs Free	BEGIN CHILDREN <A1,A2> A1 LIKE [G,.*] and CHILDREN <B1> B1 LIKE [G,.*] and CHILDREN <D1, D2> D1 LIKE [F,type,eq,Paid] D2 LIKE [F,type,eq,Free] A2 LIKE [G,.*] and CHILDREN <C1, C2> C1 LIKE [F,type,eq,Paid] C2 LIKE [F,type,eq,Free]	This query applies some two consecutive group by operations, both for paid apps and for free apps. Then, it applies another group by operation, again both for paid apps and for free apps.	PLAY STORE

Table 8: LDX Queries for Different Analysis Tasks