ORIE 3120

Prelim 1

March 5, 2019

- Closed Book / Closed Notes
- Scientific calculator permitted; using any device to access information you are expected to memorize is a violation of academic integrity.
- Points possible: 100
- Time allowed: 120 minutes.
- Write your **netid** on every page. **Students who do will be awarded 1 point**. (Exam papers may be unstapled for grading.)
- You do not need to complete all numerical calculations. For example, an answer in the form (0.1)(157) + (0.9)(123) would be just as acceptable as 126.4.
- You may bring to the exam a handwritten "formula sheet" that can have any information that you would like. This must be a single sheet of paper (front and back).
- During the exam you may only have the following items on your desk: exam, your own formula sheet, pen, pencil, eraser, calculator, photo ID, watch. Any bag you bring must stay on the floor. If you would like to get something out of your bag after the exam starts, raise your hand to ask permission of a proctor. You may not have your cell phone on your desk at any point during the exam.
- During the exam, you may not give your formula sheet to anyone else once the exam has started, nor may you accept a formula sheet from anyone else.
- If you need to go to the bathroom during the exam, raise your hand and wait for the proctor to give you permission to leave the room. Only one exam-taker will be allowed out of the exam at a time.
- Students who continue to write after the time for the exam is called will have 2 points deducted for each minute in which they are seen writing after the conclusion of the main exam time.
- Answer all questions in the space provided. Use the back of the page if necessary but do not answer on a different page unless the exam proctor provides it. Make sure each additional page contains your name. Hand in a complete exam.

Academic integrity is expected of all students of Cornell University, whether in the presence or absence of members of the faculty. Understanding this, I declare I shall not give, use, or receive unauthorized aid in this examination.

Student Signature:_____

Printed Name & Netid:_____

Problem #	Max Points	Points Awarded
Problem 1	**9**	
Problem 2	**15**	
Problem 3	**20**	
Problem 4	**20**	
Problem 5	**10**	
Problem 6	**15**	
Problem 7	**10**	
Wrote name on every page?	**1**	
Total	**100**	

Problem 1 [9 pts]

We have a database with two tables, T1 and T2. T1 has three columns (A, B, and C), and T2 has two columns (D and E).

T1

A	B	C
1	2	5
2	3	7
3	3	2
4	3	1
5	2	6

T2

D	E
1	5
2	2
3	7

For each valid foreign key relationship that may be created using this database, list in the space below the foreign key's table and field, and the referenced table and field. "Valid" means that SQLite would not generate an error if you specified this relationship. Do not include foreign key relationships from a field to itself. You do not need to fill in all of the rows in the space below.

Foreign Key Table	Foreign Key Field	Referenced Table	Referenced Field

Problem 2 [15 points]

We have a database with two tables, T1 and T2. T1 has two columns (A and B), and T2 has three columns (A, B, and C).

T1

A	B
1	apple
2	orange
3	pear
4	grape

T2

A	B	C
3	1	1
2	2	2
2	1	1
1	1	3
1	2	1
1	1	1

For each of the following queries, write the query result. Write your query result as a table, with the column names written as they would be by SQLite. For example, for the query SELECT * FROM T1, the answer would be:

A	B
1	apple
2	orange
3	pear
4	grape

Any NULLs in the response may be left as blanks.

(a) SELECT A, B, C FROM T2 WHERE A <> 2 OR B>1

(b) SELECT T1.A AS T1_A, T1.B AS T1_B, T2.A AS T2_A
FROM T1 LEFT JOIN T2 ON T1.A=T2.C
ORDER BY T1.A, T2.A

(c) SELECT A, B, SUM(C) FROM T2 GROUP BY 1, 2

Problem 3 [20 pts]

Consider a database scheme with 4 tables: PurchaseOrders, Products, ProductCategory, and DistributionCenters. This database is used by a company to track how long it takes products to be processed (sorted) when they are shipped to one of three different distribution centers. You may assume that there are no NULLs in the database.

The Products table tells us the category ID for each of 10 different products. Its primary key is ProductID. It has a foreign key whose child is Products.ProductCategoryID and parent is ProductCategory.ID.

Products

ProductID	ProductCategoryID
1	A
2	A
3	A
4	A
5	A
6	B
7	B
8	B
9	B
10	B

The ProductCategory table tells us the English name of each of the ProductCategoryIDs from the Products table. Its primary key is ID.

ProductCategory

ID	Name
A	Furniture
B	Appliances

The DistributionCenters table tells us the English name of the city in which each of the three different distribution centers is located. Its primary key is DC.

DistributionCenters

DC	Name
1	Chicago
2	Los Angeles
3	New York

The PurchaseOrders table has a record for each time a particular product was shipped to a particular distribution center. One such record is called a “purchase order”, or “PO”. It gives a unique id (the PO_Num), the ProductID of the product being shipped (where ProductID appears in the Products table), the time in days that it took to process the shipment (SortTime), and the Distribution Center (DC) to which the product was shipped. Its primary key is PO_Num. It has a foreign key whose child is PurchaseOrders.ProductID and whose parent is Products.ProductID. It has another foreign key whose child is PurchaseOrders.DC and whose parent is DistributionCenters.DC.

PurchaseOrders

PO_Num	ProductID	SortTime	DC
392	1	4	1
393	5	15	2
394	9	6	2
395	2	9	3
396	2	12	1
397	2	5	1
398	5	5	1
399	4	4	1
400	6	11	3
401	2	2	1
402	9	4	1
403	8	3	1

We want to predict, for each distribution center and product category, of what the sort time will be. To do this, we write a sequence of queries. The results can come back in any order.

a) (5 points) Write a query or queries that will calculate, for each distribution center, the average sort time for all purchase orders arriving to that distribution center. For the mock data given, the result should look like the following. Please use the same field names in your query. Below, you may refer to these query results as a view with the name Q01.

DC	AvgSortTime
1	4.875
2	10.5
3	10.0

b) (5 points) Write a query or queries that will calculate, for each distribution center and product category, the average sort time for all purchase orders from that product category arriving to that distribution center. It should also calculate the number of purchase orders in that product category arriving to that distribution center. For the mock data given, the result should look like the following. Please use the same field names in your query. Below, you may refer to these query results as a view with the name Q02.

DC	ProductCategoryID	AvgSortTime	CountSortTime
1	A	5.333333333333333	6
1	B	3.5	2
2	A	15.0	1
2	B	6.0	1
3	A	9.0	1
3	B	11.0	1

c) (5 points) Join the queries you created in parts (a) and (b) to calculate, for every pair of DC and ProductCategoryID that appear in Q02, a calculated column called PredictedSortTime. If the number of purchases orders for a pair of DC and ProductCategoryID is strictly greater than 3, then PredictedSortTime is the AvgSortTime from Q02. If not, it is AvgSortTime from Q01. For the mock data given, the result should look like the following. Please use the same field names in your query. Below, you may refer to these query results as a view with the name Q03.

DC	ProductCategoryID	PredictedSortTime
1	A	5.333333333333333
1	B	4.875
2	A	10.5
2	B	10.5
3	A	10.0
3	B	10.0

d) (5 points) Join Q03 with DistributionCenters and ProductCategory to report the same results as Q03, but with DC replaced by the English name of the distribution center, and the ProductCategoryID replaced by the English name of the product category. For the mock data given, the result should look like the following. Please use the same field names in your query.

DistributionCenterName	ProductCategoryName	PredictedSortTime
Chicago	Furniture	5.333333333333333
Chicago	Appliances	4.875
Los Angeles	Furniture	10.5
Los Angeles	Appliances	10.5
New York	Furniture	10.0
New York	Appliances	10.0

Problem 4 [20 pts]

An Ithaca-area girl scout troop is selling boxes of cookies over a weekend in March. The cookie supplier charges them $4 per box. The troop sells each box for $5. Boxes of cookies that are not sold by the end of the weekend are returned to the supplier who reimburses $3.50 for each. The troop leader estimates that demand for cookies will be normally distributed with a mean of 100 boxes and a standard deviation of 10 boxes.

a) Find the optimal number of boxes of cookies to order. Round to the nearest integer.

b) Find the sum of the expected overage and underage costs. Round to the nearest cent.

c) Find the expected profit. Round to the nearest cent.

d) What is the probability that the troop will earn at least $100 in profit if it orders 100 boxes?

e) How many boxes should the troop order to maximize the probability that they earn at least $90 in profit?

Problem 5 [10 pts]

We have a table Strings containing one field, A, with the following data:

A
aaaaaa
bbbbbb
cccccc
dddddd
abababab
acacac
adadad
bcbcbc
bdbdbd
cdcdcd
abcabc
aabbaa
aaccaa
aaddaa
bbaabb
bbccbb
bbddbb
ccaacc
ccbbcc
ccddcc

Write a query that produces the following query result. You do not need to reproduce the same order, but you do need to reproduce the field names and the field values.

A	F1	F2	F3	F4	F5	F6	F7
aaaaaa	a	aaaa	aaaaaa			AAAAAA	aaa
bbbbbb	b	bbbb	bbbbbb		bbbbbb	BBBBBB	bbb
cccccc	c	cccc	cccccc	cccccc	cccccc	CCCCCC	ccc
dddddd	d	dddd	dddddd	dddddd	dddddd	DDDDDD	ddd
ababab	a	baba	zzz		babab	ABABAB	aba
acacac	a	caca	acacac	acacac	cacac	ACACAC	aca
adadad	a	dada	adadad	adadad	dadad	ADADAD	ada
bcbcbc	b	cbcb	bcbcbc	bcbcbc	bcbcbc	BCBCBC	bcb
bdbdbd	b	dbdb	bdbdbd	bdbdbd	bdbdbd	BDBDBD	bdb
cdcdcd	c	dcdc	cdcdcd	cdcdcd	cdcdcd	CDCDCD	cdc
abcabc	a	bcab	zczc	abcabc	bcabc	ABCABC	cab
aabbaa	a	abba	azbaa		bbaa	AABBAA	bba
aaccaa	a	acca	aaccaa	aacc	ccaa	AACCAA	cca
aaddaa	a	adda	aaddaa	aadd	ddaa	AADDAA	dda
bbaabb	b	baab	bbazb		bbaabb	BBAABB	aab
bbccbb	b	bccb	bbccbb	bbcc	bbccbb	BBCCBB	ccb
bbddbb	b	bddb	bbddbb	bbdd	bbddbb	BBDDBB	ddb
ccaacc	c	caac	ccaacc	ccaacc	ccaacc	CCAACC	aac
ccbbcc	c	cbbc	ccbbcc	ccbbcc	ccbbcc	CCBBCC	bbc
ccddcc	c	cddc	ccddcc	ccddcc	ccddcc	CCDDCC	ddc

Write your query here:

Problem 6 [15 pts]

Kaiser Permanente (a large health care provider) operates a warehouse for storing and distributing medical supplies in the San Francisco Bay Area. They are evaluating how frequently they buy sterile bandages for use in medical procedures. Their supplier charges them $10 per box of bandages ordered, as well as an additional $100 per order for shipping and processing. The lead time between placing an order and receiving it is 7 days. Demand for bandages from individual health care providers serviced from the warehouse is constant over time, and is 60 boxes per day. The holding cost for a box of bandages is $10 per year. Assume there are 365 days per year.

a) Find the optimal number of boxes of bandages in each order. Round to the nearest integer.

b) Find the inventory level that would trigger a replenishment order.

c) Find the average order cost per year under the optimal order quantity, including both the cost of the bandages themselves and the shipping/processing costs. Round to the nearest cent.

Problem 7 [10 pts]

The table below describes requests for service from riders on a ridesharing platform on a single day in a single city. Each record in the table contains the time that the request was made in seconds since the start of the day, a unique id associated with that request (request_ID), and the latitude and longitude of the rider's origin. The table is called "trips".

Trips

request_ID	origin_lat	origin_lng	request_time
643219101	38.889931	-77.009003	40982.112
643219102	38.889939	-77.009104	40982.578
643219103	38.88996	-77.008901	40985.022
643219104	38.888721	-77.009523	41010.781
643219105	38.888531	-77.008923	41018.315

We refer to the number of seconds between a request and the next one in time as the *interarrival time*.

(a) Write a query (or sequence of queries) that calculates the interarrival time for each request, except the last one in the dataset. You may assume that the request_time for each request is unique. Although the order of request_ID is the same as the order of request_time in the above dataset, do not assume this in your query. You may refer to your query below as a view called Q01. Here is what your query's result would look like on the data above:

request_ID	interarrival_time
643219101	0.466
643219102	2.444
643219103	25.759
643219104	7.534

(b) Write a query or sequence of queries that calculates the average interarrival time in seconds, and the fraction of requests for which the interarrival time is strictly above 1 second. Your query should work regardless of the number of records in trips. Your query should produce the following result on the data above:

Avg_Interarrival_Time	Fraction_Above_1
9.05075	0.75

Table values the density f(z) of a normal random variable with mean 0 and variance 1, where z is the value at left plus the value at top, e.g. f(3.99) = 0.00014

Z	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.39894	0.39892	0.39886	0.39876	0.39862	0.39844	0.39822	0.39797	0.39767	0.39733
0.1	0.39695	0.39654	0.39608	0.39559	0.39505	0.39448	0.39387	0.39322	0.39253	0.39181
0.2	0.39104	0.39024	0.38940	0.38853	0.38762	0.38667	0.38568	0.38466	0.38361	0.38251
0.3	0.38139	0.38023	0.37903	0.37780	0.37654	0.37524	0.37391	0.37255	0.37115	0.36973
0.4	0.36827	0.36678	0.36526	0.36371	0.36213	0.36053	0.35889	0.35723	0.35553	0.35381
0.5	0.35207	0.35029	0.34849	0.34667	0.34482	0.34294	0.34105	0.33912	0.33718	0.33521
0.6	0.33322	0.33121	0.32918	0.32713	0.32506	0.32297	0.32086	0.31874	0.31659	0.31443
0.7	0.31225	0.31006	0.30785	0.30563	0.30339	0.30114	0.29887	0.29659	0.29431	0.29200
0.8	0.28969	0.28737	0.28504	0.28269	0.28034	0.27798	0.27562	0.27324	0.27086	0.26848
0.9	0.26609	0.26369	0.26129	0.25888	0.25647	0.25406	0.25164	0.24923	0.24681	0.24439
1.0	0.24197	0.23955	0.23713	0.23471	0.23230	0.22988	0.22747	0.22506	0.22265	0.22025
1.1	0.21785	0.21546	0.21307	0.21069	0.20831	0.20594	0.20357	0.20121	0.19886	0.19652
1.2	0.19419	0.19186	0.18954	0.18724	0.18494	0.18265	0.18037	0.17810	0.17585	0.17360
1.3	0.17137	0.16915	0.16694	0.16474	0.16256	0.16038	0.15822	0.15608	0.15395	0.15183
1.4	0.14973	0.14764	0.14556	0.14350	0.14146	0.13943	0.13742	0.13542	0.13344	0.13147
1.5	0.12952	0.12758	0.12566	0.12376	0.12188	0.12001	0.11816	0.11632	0.11450	0.11270
1.6	0.11092	0.10915	0.10741	0.10567	0.10396	0.10226	0.10059	0.09893	0.09728	0.09566
1.7	0.09405	0.09246	0.09089	0.08933	0.08780	0.08628	0.08478	0.08329	0.08183	0.08038
1.8	0.07895	0.07754	0.07614	0.07477	0.07341	0.07206	0.07074	0.06943	0.06814	0.06687
1.9	0.06562	0.06438	0.06316	0.06195	0.06077	0.05959	0.05844	0.05730	0.05618	0.05508
2.0	0.05399	0.05292	0.05186	0.05082	0.04980	0.04879	0.04780	0.04682	0.04586	0.04491
2.1	0.04398	0.04307	0.04217	0.04128	0.04041	0.03955	0.03871	0.03788	0.03706	0.03626
2.2	0.03547	0.03470	0.03394	0.03319	0.03246	0.03174	0.03103	0.03034	0.02965	0.02898
2.3	0.02833	0.02768	0.02705	0.02643	0.02582	0.02522	0.02463	0.02406	0.02349	0.02294
2.4	0.02239	0.02186	0.02134	0.02083	0.02033	0.01984	0.01936	0.01888	0.01842	0.01797
2.5	0.01753	0.01709	0.01667	0.01625	0.01585	0.01545	0.01506	0.01468	0.01431	0.01394
2.6	0.01358	0.01323	0.01289	0.01256	0.01223	0.01191	0.01160	0.01130	0.01100	0.01071
2.7	0.01042	0.01014	0.00987	0.00961	0.00935	0.00909	0.00885	0.00861	0.00837	0.00814
2.8	0.00792	0.00770	0.00748	0.00727	0.00707	0.00687	0.00668	0.00649	0.00631	0.00613
2.9	0.00595	0.00578	0.00562	0.00545	0.00530	0.00514	0.00499	0.00485	0.00470	0.00457
3.0	0.00443	0.00430	0.00417	0.00405	0.00393	0.00381	0.00370	0.00358	0.00348	0.00337
3.1	0.00327	0.00317	0.00307	0.00298	0.00288	0.00279	0.00271	0.00262	0.00254	0.00246
3.2	0.00238	0.00231	0.00224	0.00216	0.00210	0.00203	0.00196	0.00190	0.00184	0.00178
3.3	0.00172	0.00167	0.00161	0.00156	0.00151	0.00146	0.00141	0.00136	0.00132	0.00127
3.4	0.00123	0.00119	0.00115	0.00111	0.00107	0.00104	0.00100	0.00097	0.00094	0.00090
3.5	0.00087	0.00084	0.00081	0.00079	0.00076	0.00073	0.00071	0.00068	0.00066	0.00063
3.6	0.00061	0.00059	0.00057	0.00055	0.00053	0.00051	0.00049	0.00047	0.00046	0.00044
3.7	0.00042	0.00041	0.00039	0.00038	0.00037	0.00035	0.00034	0.00033	0.00031	0.00030
3.8	0.00029	0.00028	0.00027	0.00026	0.00025	0.00024	0.00023	0.00022	0.00021	0.00021
3.9	0.00020	0.00019	0.00018	0.00018	0.00017	0.00016	0.00016	0.00015	0.00014	0.00014

Table values are P(Z<=z), where z is the value at left plus the value at top, e.g. P(Z<=3.99) = .99997										
Z	**0**	**0.01**	**0.02**	**0.03**	**0.04**	**0.05**	**0.06**	**0.07**	**0.08**	**0.09**
0.0	0.50000	0.50399	0.50798	0.51197	0.51595	0.51994	0.52392	0.52790	0.53188	0.53586
0.1	0.53983	0.54380	0.54776	0.55172	0.55567	0.55962	0.56356	0.56749	0.57142	0.57535
0.2	0.57926	0.58317	0.58706	0.59095	0.59483	0.59871	0.60257	0.60642	0.61026	0.61409
0.3	0.61791	0.62172	0.62552	0.62930	0.63307	0.63683	0.64058	0.64431	0.64803	0.65173
0.4	0.65542	0.65910	0.66276	0.66640	0.67003	0.67364	0.67724	0.68082	0.68439	0.68793
0.5	0.69146	0.69497	0.69847	0.70194	0.70540	0.70884	0.71226	0.71566	0.71904	0.72240
0.6	0.72575	0.72907	0.73237	0.73565	0.73891	0.74215	0.74537	0.74857	0.75175	0.75490
0.7	0.75804	0.76115	0.76424	0.76730	0.77035	0.77337	0.77637	0.77935	0.78230	0.78524
0.8	0.78814	0.79103	0.79389	0.79673	0.79955	0.80234	0.80511	0.80785	0.81057	0.81327
0.9	0.81594	0.81859	0.82121	0.82381	0.82639	0.82894	0.83147	0.83398	0.83646	0.83891
1.0	0.84134	0.84375	0.84614	0.84849	0.85083	0.85314	0.85543	0.85769	0.85993	0.86214
1.1	0.86433	0.86650	0.86864	0.87076	0.87286	0.87493	0.87698	0.87900	0.88100	0.88298
1.2	0.88493	0.88686	0.88877	0.89065	0.89251	0.89435	0.89617	0.89796	0.89973	0.90147
1.3	0.90320	0.90490	0.90658	0.90824	0.90988	0.91149	0.91309	0.91466	0.91621	0.91774
1.4	0.91924	0.92073	0.92220	0.92364	0.92507	0.92647	0.92785	0.92922	0.93056	0.93189
1.5	0.93319	0.93448	0.93574	0.93699	0.93822	0.93943	0.94062	0.94179	0.94295	0.94408
1.6	0.94520	0.94630	0.94738	0.94845	0.94950	0.95053	0.95154	0.95254	0.95352	0.95449
1.7	0.95543	0.95637	0.95728	0.95818	0.95907	0.95994	0.96080	0.96164	0.96246	0.96327
1.8	0.96407	0.96485	0.96562	0.96638	0.96712	0.96784	0.96856	0.96926	0.96995	0.97062
1.9	0.97128	0.97193	0.97257	0.97320	0.97381	0.97441	0.97500	0.97558	0.97615	0.97670
2.0	0.97725	0.97778	0.97831	0.97882	0.97932	0.97982	0.98030	0.98077	0.98124	0.98169
2.1	0.98214	0.98257	0.98300	0.98341	0.98382	0.98422	0.98461	0.98500	0.98537	0.98574
2.2	0.98610	0.98645	0.98679	0.98713	0.98745	0.98778	0.98809	0.98840	0.98870	0.98899
2.3	0.98928	0.98956	0.98983	0.99010	0.99036	0.99061	0.99086	0.99111	0.99134	0.99158
2.4	0.99180	0.99202	0.99224	0.99245	0.99266	0.99286	0.99305	0.99324	0.99343	0.99361
2.5	0.99379	0.99396	0.99413	0.99430	0.99446	0.99461	0.99477	0.99492	0.99506	0.99520
2.6	0.99534	0.99547	0.99560	0.99573	0.99585	0.99598	0.99609	0.99621	0.99632	0.99643
2.7	0.99653	0.99664	0.99674	0.99683	0.99693	0.99702	0.99711	0.99720	0.99728	0.99736
2.8	0.99744	0.99752	0.99760	0.99767	0.99774	0.99781	0.99788	0.99795	0.99801	0.99807
2.9	0.99813	0.99819	0.99825	0.99831	0.99836	0.99841	0.99846	0.99851	0.99856	0.99861
3.0	0.99865	0.99869	0.99874	0.99878	0.99882	0.99886	0.99889	0.99893	0.99896	0.99900
3.1	0.99903	0.99906	0.99910	0.99913	0.99916	0.99918	0.99921	0.99924	0.99926	0.99929
3.2	0.99931	0.99934	0.99936	0.99938	0.99940	0.99942	0.99944	0.99946	0.99948	0.99950
3.3	0.99952	0.99953	0.99955	0.99957	0.99958	0.99960	0.99961	0.99962	0.99964	0.99965
3.4	0.99966	0.99968	0.99969	0.99970	0.99971	0.99972	0.99973	0.99974	0.99975	0.99976
3.5	0.99977	0.99978	0.99978	0.99979	0.99980	0.99981	0.99981	0.99982	0.99983	0.99983
3.6	0.99984	0.99985	0.99985	0.99986	0.99986	0.99987	0.99987	0.99988	0.99988	0.99989
3.7	0.99989	0.99990	0.99990	0.99990	0.99991	0.99991	0.99992	0.99992	0.99992	0.99992
3.8	0.99993	0.99993	0.99993	0.99994	0.99994	0.99994	0.99994	0.99995	0.99995	0.99995
3.9	0.99995	0.99995	0.99996	0.99996	0.99996	0.99996	0.99996	0.99996	0.99997	0.99997