SQL

Schwartz

December 6, 2016

Sizes of things

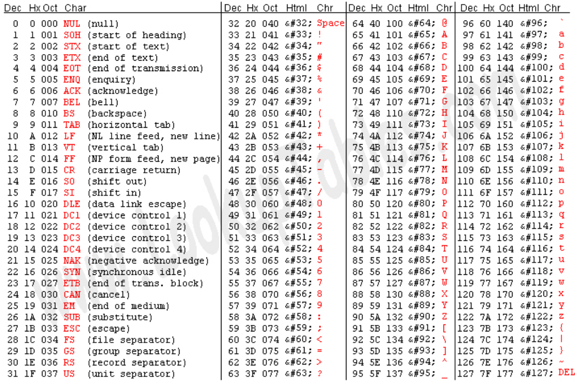
Name	Binary		Decimal		Example
	Power	Value	Power	Value	
Bit	2^{0}	1			Binary (0 or 1)
Byte (B)	2^{3}	8			"S" $=$ 01010011
Kilobyte (KB)	2^{10}	1,024	10^{3}	1,000	Word Document
Megabyte (MB)	2^{20}	1,048,567	10^{6}	1,000,000	Digital Photo
Gigabyte (GB)	2^{30}	1,073,741,824	10^{9}	1,000,000,000	DVD
Terabyte (TB)	2^{40}	1,099,511,627,776	2^{12}	1,000,000,000,000	Hard Drive
Petabyte (PB)	2^{50}	1,125,899,906,842,624	2^{15}	1,000,000,000,000,000	Some of Facebook
All Atoms	2^{266}	$\cdots$	10^{80}	$\cdots$	Universe
TSP routes	2^{329}	$(71-1)!/2$	10^{99}	$\cdots$	71 cities

Sizes of things

Name	Binary Power	Value	Decimal Power	Value	Example
Bit	2^0	1			Binary (0 or 1)
Byte (B)	2^3	8			"S" = 01010011
Kilobyte (KB)	2^{10}	1,024	10^3	1,000	Word Document
Megabyte (MB)	2^{20}	1,048,567	10^6	1,000,000	Digital Photo
Gigabyte (GB)	2^{30}	1,073,741,824	10^9	1,000,000,000	DVD
Terabyte (TB)	2^{40}	1,099,511,627,776	2^{12}	1,000,000,000,000	Hard Drive
Petabyte (PB)	2^{50}	1,125,899,906,842,624	2^{15}	1,000,000,000,000,000	Some of Facebook
All Atoms	2^{266}	$\cdots$	10^{80}	$\cdots$	Universe
TSP routes	2^{329}	$(71-1)!/2$	10^{99}	$\cdots$	71 cities

$$\textit{Ascii} = \sum_{i=1}^{8} b_i 2^{i-1}$$

$$b_i \in \{0,1\}$$



Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH	(start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX	(start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX	(end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT	(end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ	(enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK	(acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL	(bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS	(backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB	(horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT	(vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR	(carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO	(shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI	(shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE	(data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1	(device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2	(device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3	(device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4	(device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK	(negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN	(synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB	(end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM	(end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB	(substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC	(escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS	(file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS	(group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS	(record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US	(unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Objectives

1. Learn some SQL
   - SELECT
   - AS, DISTINCT
   - *, /, +, -,
   - CONCAT, ROUND, CAST, COALESCE
   - CASE WHEN THEN ELSE END
   - FROM/JOIN ON
   - LEFT, RIGHT
   - WHERE
   - AND, OR, BETWEEN, LIKE, IN, IS NULL
   - GROUP BY
   - MAX, MIN, SUM, AVG, COUNT
   - HAVING
   - (SELECT ...)
   - ORDER BY/LIMIT
2. Practice, practice, practice...

Relational Database Management System (RDBMS)

- *Persistance*: non-volatile storage
- *ACID*: reliability properties
- *Schema*: tables and typed data columns
- *Keys*: data relationships

Relational Database Management System (RDBMS)

- *Persistance*: non-volatile storage
- *ACID*: reliability properties
- *Schema*: tables and typed data columns
- *Keys*: data relationships

- Efficient queries of data and relations therein

ACID

Transactions in an RDBMS follow the *ACID* principles:

ACID

Transactions in an RDBMS follow the *ACID* principles:

A: Atomicity – “all or nothing”

ACID

Transactions in an RDBMS follow the *ACID* principles:

A: Atomicity – “all or nothing”

C: Consistency – “remain in legal state”

ACID

Transactions in an RDBMS follow the *ACID* principles:

A: Atomicity – “all or nothing”

C: Consistency – “remain in legal state”

I: Isolation – “appropriate independence”

ACID

Transactions in an RDBMS follow the *ACID* principles:

A: Atomicity – “all or nothing”

C: Consistency – “remain in legal state”

I: Isolation – “appropriate independence”

D: Durability – “persistance”

Schema

```
CREATE TABLE users {
    id INTEGER PRIMARY KEY,
    name VARCHAR(255),
    age INTEGER,
    city VARCHAR(255),
    name VARCHAR(2)
}
```

Schema

```
CREATE TABLE users {
    id INTEGER PRIMARY KEY,
    name VARCHAR(255),
    age INTEGER,
    city VARCHAR(255),
    name VARCHAR(2)
}
```

- Whitespace doesn't matter

Schema

```
CREATE TABLE users {
    id INTEGER PRIMARY KEY,
    name VARCHAR(255),
    age INTEGER,
    city VARCHAR(255),
    name VARCHAR(2)
}
```

- Whitespace doesn't matter
  (but it can help make code clearer)

Schema

```
CREATE TABLE users {
    id INTEGER PRIMARY KEY,
    name VARCHAR(255),
    age INTEGER,
    city VARCHAR(255),
    name VARCHAR(2)
}
```

- Whitespace doesn't matter

  (but it can help make code clearer)
- Capitalization (often) doesn't matter

Schema

```
CREATE TABLE users {
    id INTEGER PRIMARY KEY,
    name VARCHAR(255),
    age INTEGER,
    city VARCHAR(255),
    name VARCHAR(2)
}
```

- Whitespace doesn't matter

  (but it can help make code clearer)
- Capitalization (often) doesn't matter

  (but it can help make code clearer)

Schema

```
CREATE TABLE users {
    id INTEGER PRIMARY KEY,
    name VARCHAR(255),
    age INTEGER,
    city VARCHAR(255),
    name VARCHAR(2)
}
```

- Whitespace doesn't matter

  (but it can help make code clearer)
- Capitalization (often) doesn't matter

  (but it can help make code clearer)
- Don't look like a noob
  - follow ubiquitous conventions
  - write beautiful looking code

Schema *efficiency*

```
CREATE TABLE visits {
    id INTEGER PRIMARY KEY,
    created_at TIMESTAMP,
    user_id INTEGER REFERENCES users(id)
    -- place foreign keys on the "many"
    -- side of a one-to-many relationship
}
```

```
CREATE TABLE posts {
    id INTEGER PRIMARY KEY,
    title VARCHAR(255)
}
```

```
CREATE TABLE tags {
    id INTEGER PRIMARY KEY,
    tag VARCHAR(255)
}
```

```
CREATE TABLE posts_tags {
    post_id INTEGER REFERENCES posts(id),
    tag_id INTEGER REFERENCES tags(id)
    -- "Normalized" data only duplicates foreign keys
}
```

Structured Query Language (SQL)

SQL is used to interact with RDBMS, allowing one to

- create tables (we saw this previously)
- alter tables
- insert records
- update records
- delete records
- query records within and across tables

Structured Query Language (SQL)

SQL is used to interact with RDBMS, allowing one to

- **create tables** (we saw this previously)
- alter tables
- insert records
- update records
- delete records
- query records within and across tables

```
CREATE [TEMPORARY] TABLE table AS <SQL query>;
```

Structured Query Language (SQL)

SQL is used to interact with RDBMS, allowing one to

- create tables (we saw this previously)
- **alter tables**
- insert records
- update records
- delete records
- query records within and across tables

```
ALTER TABLE table [DROP/ADD/ALTER] column [datatype];
```

Structured Query Language (SQL)

SQL is used to interact with RDBMS, allowing one to

- create tables (we saw this previously)
- **alter tables**
- insert records
- update records
- delete records
- query records within and across tables

```
DROP TABLE table;
```

Structured Query Language (SQL)

SQL is used to interact with RDBMS, allowing one to

- create tables (we saw this previously)
- alter tables
- **insert records**
- update records
- delete records
- query records within and across tables

```
INSERT INTO table [(c1,c2,c3,...)] VALUES (v1,v2,v3,...);
```

Structured Query Language (SQL)

SQL is used to interact with RDBMS, allowing one to

- create tables (we saw this previously)
- alter tables
- insert records
- **update records**
- delete records
- query records within and across tables

```
UPDATE table SET c1=v1,c2=v2,...WHERE cX=vX;
```

Structured Query Language (SQL)

SQL is used to interact with RDBMS, allowing one to

- create tables (we saw this previously)
- alter tables
- insert records
- update records
- **delete records**
- query records within and across tables

```
DELETE FROM table WHERE cX=vX;
```

Structured Query Language (SQL)

SQL is used to interact with RDBMS, allowing one to

- create tables (we saw this previously)
- alter tables
- insert records
- update records
- delete records
- **query records within and across tables**

```
SELECT.FROM.JOIN.ON.WHERE.GROUP BY.HAVING.ORDER BY.LIMIT
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

SELECT

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

SELECT *

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

SELECT *

  FROM

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT *

    FROM table
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,

    FROM table
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),'%')

    FROM table
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),'%'),
    CASE

    FROM table
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),'%'),
    CASE WHEN

    FROM table
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),'%'),
    CASE WHEN –

    FROM table
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),'%'),
    CASE WHEN –THEN

    FROM table
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’

    FROM table
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’

    FROM table
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END
    FROM table
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS
    FROM table
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),'%'),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table AS t
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat

    FROM table t
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat

    FROM table t JOIN
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),'%'),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t JOIN table2
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),'%'),
    CASE WHEN – THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat

    FROM table t JOIN table2 AS t2
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN – THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t JOIN table2 t2
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat

    FROM table t JOIN table2 t2 ON
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t JOIN table2 t2 ON (table.id = table2.id2)
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
   CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
   FROM table t JOIN table2 t2 ON (t.id = t2.id2)
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),'%'),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t JOIN table2 t2 ON (id = id2)
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),'%'),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t JOIN table2 t2 ON id = id2
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t JOIN table2 t2 ON (t.id = t2.id2)
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),'%'),
  CASE WHEN –THEN 'a' WHEN – THEN 'b' ELSE 'c' END AS cat
  FROM table t JOIN table2 t2 ON (t.id = t2.id2)
    WHERE
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),'%'),
  CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
  FROM table t JOIN table2 t2 ON (t.id = t2.id2)
    WHERE t.c4
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),'%'),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t JOIN table2 t2 ON (t.id = t2.id2)
        WHERE t.c4<=70
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t JOIN table2 t2 ON (t.id = t2.id2)
        WHERE t.c4<=70 AND
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),'%'),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t JOIN table2 t2 ON (t.id = t2.id2)
        WHERE t.c4<=70 OR
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),'%'),
  CASE WHEN – THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
  FROM table t JOIN table2 t2 ON (t.id = t2.id2)
    WHERE t.c4<=70 OR t2.c4
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t JOIN table2 t2 ON (t.id = t2.id2)
        WHERE t.c4<=70 OR t2.c4 LIKE
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
   CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
   FROM table t JOIN table2 t2 ON (t.id = t2.id2)
      WHERE t.c4<=70 OR t2.c4 LIKE ‘S%’
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),'%'),
  CASE WHEN – THEN 'a' WHEN – THEN 'b' ELSE 'c' END AS cat
  FROM table t JOIN table2 t2 ON (t.id = t2.id2)
    WHERE (t.c4<=70 OR t2.c4 LIKE 'S%')
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t JOIN table2 t2 ON (t.id = t2.id2)
        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t JOIN table2 t2 ON (t.id = t2.id2)
        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND cat
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t JOIN table2 t2 ON (t.id = t2.id2)
        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND cat IN
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),'%'),
  CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
  FROM table t JOIN table2 t2 ON (t.id = t2.id2)
    WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND cat IN (‘a’,‘c’)
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t LEFT OUTER JOIN table2 t2 ON (t.id = t2.id2)
        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND cat IN (‘a’,‘c’)
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)
        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND cat IN (‘a’,‘c’)
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)
        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                   cat IN (‘a’,‘c’)
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)
        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                   cat IN (‘a’,‘c’) OR
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)
        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                    cat IN (‘a’,‘c’) OR t2.id2
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)
        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                   cat IN (‘a’,‘c’) OR t2.id2 IS NULL
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN – THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)
        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat

    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)

        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
              (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)

        GROUP BY
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT     c2,CONCAT(ROUND(100*c3/CAST(c2 AS REAL),2),’%’),
    CASE WHEN – THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat

    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)

        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                   (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)

        GROUP BY c2, cat
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT CONCAT(MIN(ROUND(100*c3/CAST(c2 AS REAL),2)),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)
        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                 (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)
        GROUP BY c2, cat
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT CONCAT(MAX(ROUND(100*c3/CAST(c2 AS REAL),2)),'%'),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat

    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)

        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                    (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)

        GROUP BY c2, cat
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT CONCAT(MAX(ROUND(100*c3/CAST(c2 AS REAL),2)),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)
        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                   (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)
        GROUP BY c2, cat
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT CONCAT(AVG(ROUND(100*c3/CAST(c2 AS REAL),2)),’%’),
    CASE WHEN – THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat

    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)

        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                   (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)

        GROUP BY c2, cat
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT c1,CONCAT(AVG(ROUND(100*c3/CAST(c2 AS REAL),2)),’%
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat

    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)

        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                    (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)

        GROUP BY c2, cat
        HAVING
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT CONCAT(AVG(ROUND(100*c3/CAST(c2 AS REAL),2)),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat

    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)

        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                   (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)

        GROUP BY c2, cat
        HAVING AVE(1)
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT CONCAT(AVG(ROUND(100*c3/CAST(c2 AS REAL),2)),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat

    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)

        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                   (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)

        GROUP BY c2, cat
        HAVING AVE(c1)
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT CONCAT(AVG(ROUND(100*c3/CAST(c2 AS REAL),2)),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)
        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                   (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)
        GROUP BY c2, cat
        HAVING AVE(c1) >
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT CONCAT(AVG(ROUND(100*c3/CAST(c2 AS REAL),2)),'%'),
    CASE WHEN – THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat

    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)

        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
              (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)

        GROUP BY c2, cat
        HAVING AVE(c1) > ()
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT CONCAT(AVG(ROUND(100*c3/CAST(c2 AS REAL),2)),’%’),
    CASE WHEN – THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)
        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                   (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)
        GROUP BY c2, cat
        HAVING AVE(c1) > (SELECT DISTINCT COUNT(*) FROM t3
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT CONCAT(AVG(ROUND(100*c3/CAST(c2 AS REAL),2)),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat

    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)

        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                   (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)

        GROUP BY c2, cat
        HAVING AVE(c1) > (SELECT DISTINCT COUNT(1) FROM t3
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT CONCAT(AVG(ROUND(100*c3/CAST(c2 AS REAL),2)),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)
        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                 (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)
        GROUP BY c2, cat
        HAVING AVE(c1) > (SELECT COUNT(DISTINCT c5) FROM t3
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT CONCAT(AVG(ROUND(100*c3/CAST(c2 AS REAL),2)),’%’),
    CASE WHEN – THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat
    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)
        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                   (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)
        GROUP BY c2, cat
        HAVING AVE(c1) > (SELECT COUNT(DISTINCT c5) FROM t3
                                     WHERE c5 BETWEEN ‘J’ AND ‘M’)
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT CONCAT(AVG(ROUND(100*c3/CAST(c2 AS REAL),2)),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat

    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)

        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                  (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)

        GROUP BY c2, cat
        HAVING AVE(c1) > (SELECT COUNT(DISTINCT c5) FROM t3
                              WHERE c5 BETWEEN ‘J’ AND ‘M’)
    ORDER BY
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT CONCAT(AVG(ROUND(100*c3/CAST(c2 AS REAL),2)),’%’),
    CASE WHEN – THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat

    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)

        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
              (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)

        GROUP BY c2, cat
        HAVING AVE(c1) > (SELECT COUNT(DISTINCT c5) FROM t3
                              WHERE c5 BETWEEN ‘J’ AND ‘M’)
    ORDER BY cat
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT CONCAT(AVG(ROUND(100*c3/CAST(c2 AS REAL),2)),'%'),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat

    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)

        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                   (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)

        GROUP BY c2, cat
        HAVING AVE(c1) > (SELECT COUNT(DISTINCT c5) FROM t3
                                         WHERE c5 BETWEEN ‘J’ AND ‘M’)
    ORDER BY cat
    LIMIT 1
```

Declarative Language: *say what – not how*

The details of how things are actually done is just left up to SQL

Anatomy of a query:

```
SELECT CONCAT(AVG(ROUND(100*c3/CAST(c2 AS REAL),2)),’%’),
    CASE WHEN –THEN ‘a’ WHEN – THEN ‘b’ ELSE ‘c’ END AS cat

    FROM table t LEFT JOIN table2 t2 ON (t.id = t2.id2)

        WHERE (t.c4<=70 OR t2.c4 LIKE ‘S%’) AND
                   (cat IN (‘a’,‘c’) OR t2.id2 IS NULL)

        GROUP BY c2, cat
        HAVING AVE(c1) > (SELECT COUNT(DISTINCT c5) FROM t3
                               WHERE c5 BETWEEN ‘J’ AND ‘M’)
    ORDER BY cat


    LIMIT 1;
```

SQL *order of operations*

SQL *order of operations*

1. FROM/JOIN ON

SQL *order of operations*

1. FROM/JOIN ON
2. WHERE

SQL *order of operations*

1. FROM/JOIN ON
2. WHERE
3. GROUP BY

SQL *order of operations*

1. FROM/JOIN ON
2. WHERE
3. GROUP BY
4. HAVING

SQL *order of operations*

1. FROM/JOIN ON
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT

SQL *order of operations*

1. FROM/JOIN ON
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT
6. DISTINCT

SQL *order of operations*

1. FROM/JOIN ON
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT
6. DISTINCT
7. ORDER BY/LIMIT

Conclusion (and SUPER HINT)

It doesn't cost anything to

CREATE TABLE table AS (SELECT ...)

use it, and then

DROP TABLE table