TECH
VAULT

CARDINALITY ESTIMATION PART (1)

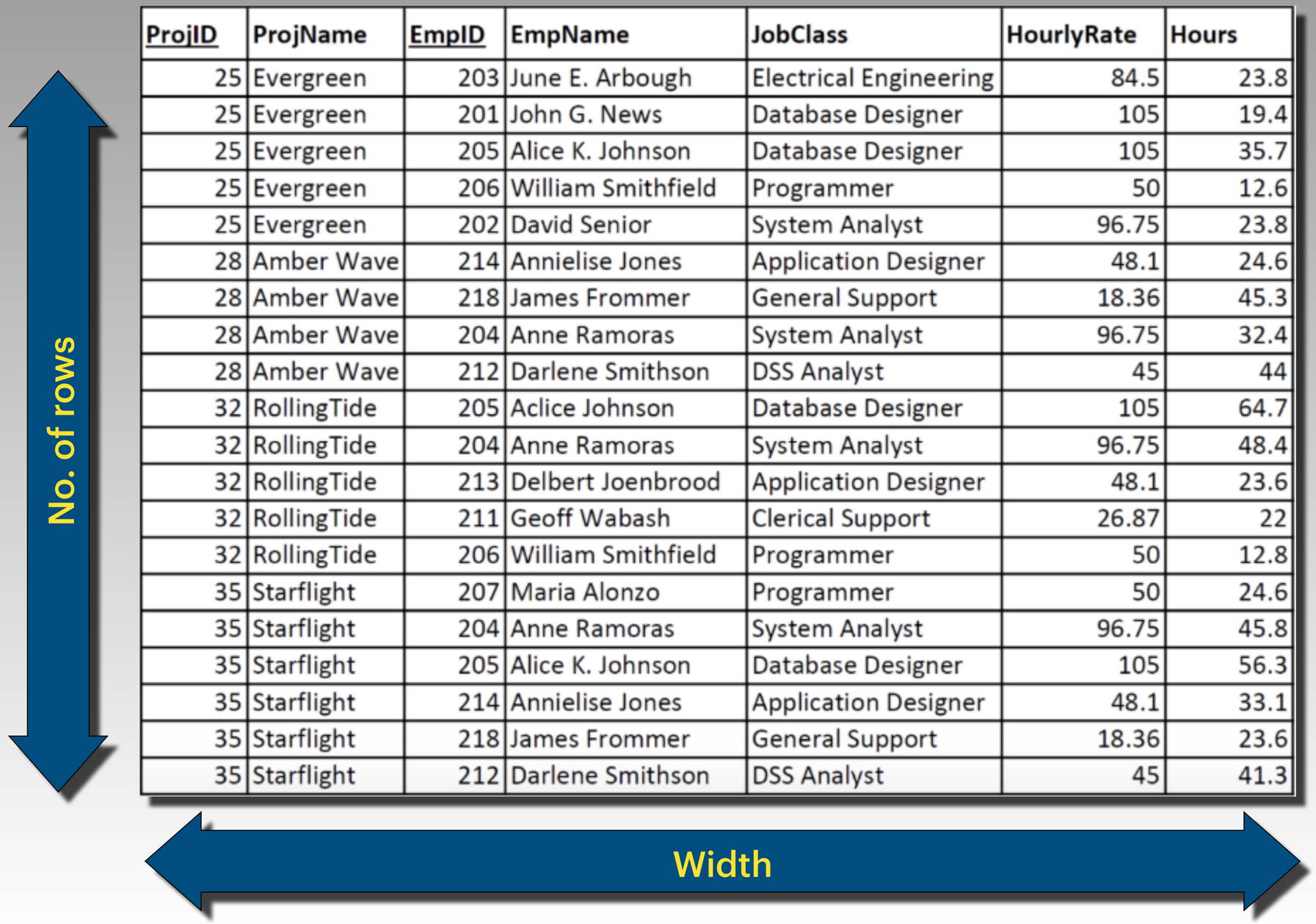
AMR ELHELW

```
EXPLAIN SELECT e.id, e.name FROM employee e JOIN dept d ON e.dept_id = d.id
WHERE d.name='Engineering' ORDER BY e.name;

                                QUERY PLAN
-------------------------------------------------------------------------
 Sort  (cost=114.47..115.09 rows=250 width=17)
   Sort Key: e.name
   ->  Hash Join  (cost=1.26..104.51 rows=250 width=17)
         Hash Cond: (e.dept_id = d.id)
         ->  Seq Scan on employee e  (cost=0.00..82.00 rows=5000 width=21)
         ->  Hash  (cost=1.25..1.25 rows=1 width=4)
               ->  Seq Scan on dept d  (cost=0.00..1.25 rows=1 width=4)
                     Filter: ((name)::text = 'Engineering'::text)
(8 rows)
```

postgres / src / include / optimizer / **cost.h**

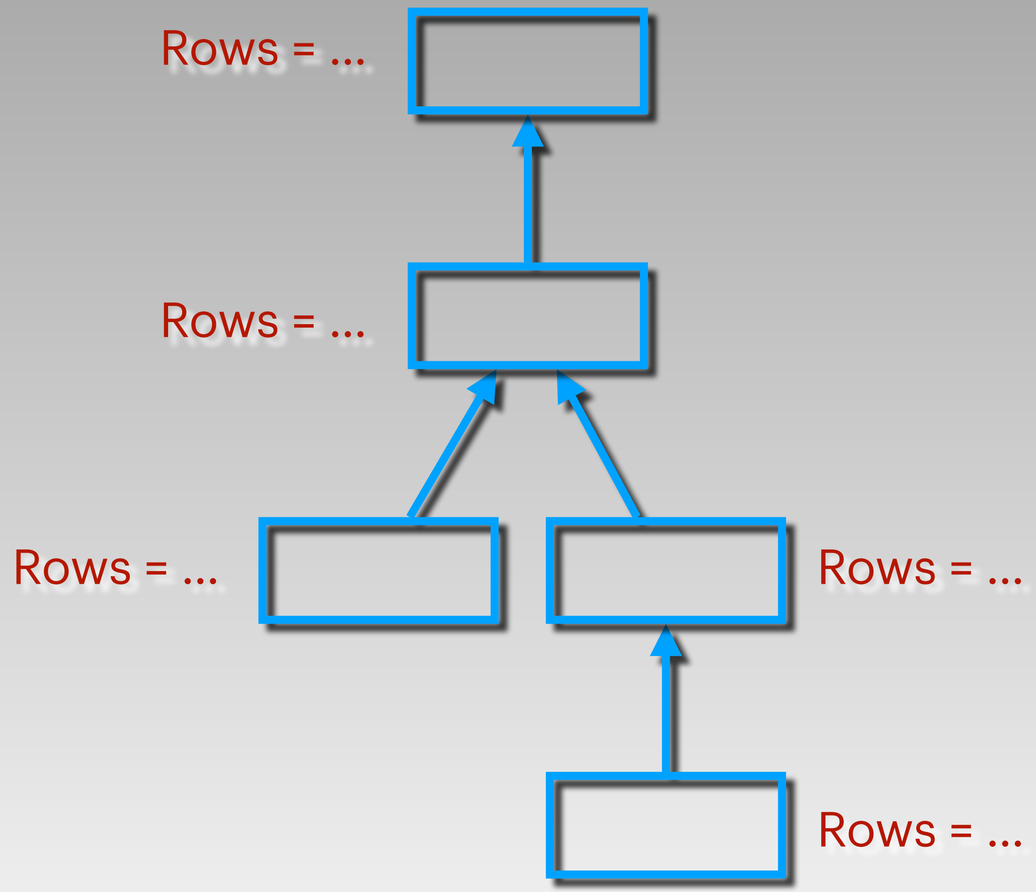
```
extern void cost_seqscan(Path *path, PlannerInfo *root, RelOptInfo *baserel,
                         ParamPathInfo *param_info);
extern void cost_samplescan(Path *path, PlannerInfo *root, RelOptInfo *baserel,
                            ParamPathInfo *param_info);
extern void cost_index(IndexPath *path, PlannerInfo *root,
                       double loop_count, bool partial_path);
extern void cost_bitmap_heap_scan(Path *path, PlannerInfo *root, RelOptInfo *baserel,
                                  ParamPathInfo *param_info,
                                  Path *bitmapqual, double loop_count);
extern void cost_bitmap_and_node(BitmapAndPath *path, PlannerInfo *root);
extern void cost_bitmap_or_node(BitmapOrPath *path, PlannerInfo *root);
extern void cost_bitmap_tree_node(Path *path, Cost *cost, Selectivity *selec);
extern void cost_tidscan(Path *path, PlannerInfo *root,
                         RelOptInfo *baserel, List *tidquals, ParamPathInfo *param_info);
extern void cost_tidrangescan(Path *path, PlannerInfo *root,
                              RelOptInfo *baserel, List *tidrangequals,
                              ParamPathInfo *param_info);
extern void cost_subqueryscan(SubqueryScanPath *path, PlannerInfo *root,
                              RelOptInfo *baserel, ParamPathInfo *param_info,
                              bool trivial_pathtarget);
extern void cost_functionscan(Path *path, PlannerInfo *root,
                              RelOptInfo *baserel, ParamPathInfo *param_info);
extern void cost_valuesscan(Path *path, PlannerInfo *root,
                            RelOptInfo *baserel, ParamPathInfo *param_info);
extern void cost_tablefuncscan(Path *path, PlannerInfo *root,
                               RelOptInfo *baserel, ParamPathInfo *param_info);
extern void cost_ctescan(Path *path, PlannerInfo *root,
                         RelOptInfo *baserel, ParamPathInfo *param_info);
extern void cost_namedtuplestorescan(Path *path, PlannerInfo *root,
                                     RelOptInfo *baserel, ParamPathInfo *param_info);
```



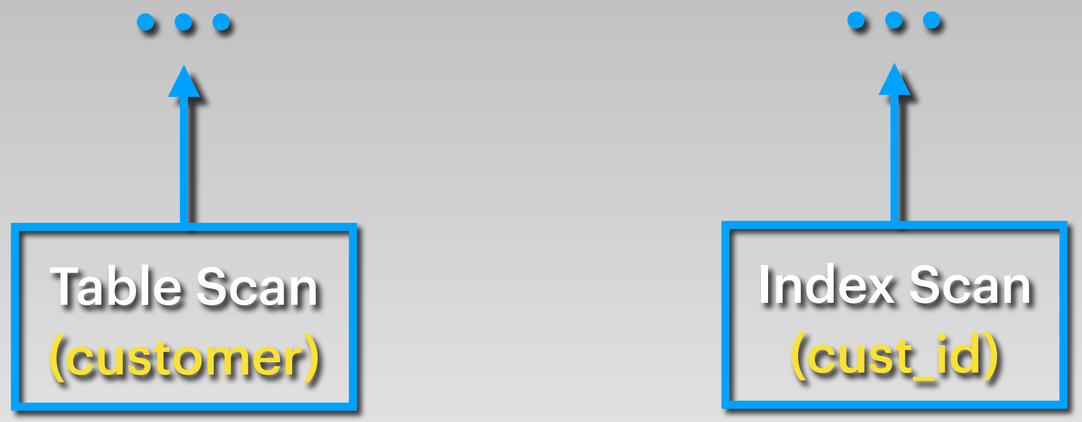
ProjID	ProjName	EmpID	EmpName	JobClass	HourlyRate	Hours
25	Evergreen	203	June E. Arbough	Electrical Engineering	84.5	23.8
25	Evergreen	201	John G. News	Database Designer	105	19.4
25	Evergreen	205	Alice K. Johnson	Database Designer	105	35.7
25	Evergreen	206	William Smithfield	Programmer	50	12.6
25	Evergreen	202	David Senior	System Analyst	96.75	23.8
28	Amber Wave	214	Annielise Jones	Application Designer	48.1	24.6
28	Amber Wave	218	James Frommer	General Support	18.36	45.3
28	Amber Wave	204	Anne Ramoras	System Analyst	96.75	32.4
28	Amber Wave	212	Darlene Smithson	DSS Analyst	45	44
32	RollingTide	205	Aclice Johnson	Database Designer	105	64.7
32	RollingTide	204	Anne Ramoras	System Analyst	96.75	48.4
32	RollingTide	213	Delbert Joenbrood	Application Designer	48.1	23.6
32	RollingTide	211	Geoff Wabash	Clerical Support	26.87	22
32	RollingTide	206	William Smithfield	Programmer	50	12.8
35	Starflight	207	Maria Alonzo	Programmer	50	24.6
35	Starflight	204	Anne Ramoras	System Analyst	96.75	45.8
35	Starflight	205	Alice K. Johnson	Database Designer	105	56.3
35	Starflight	214	Annielise Jones	Application Designer	48.1	33.1
35	Starflight	218	James Frommer	General Support	18.36	23.6
35	Starflight	212	Darlene Smithson	DSS Analyst	45	41.3

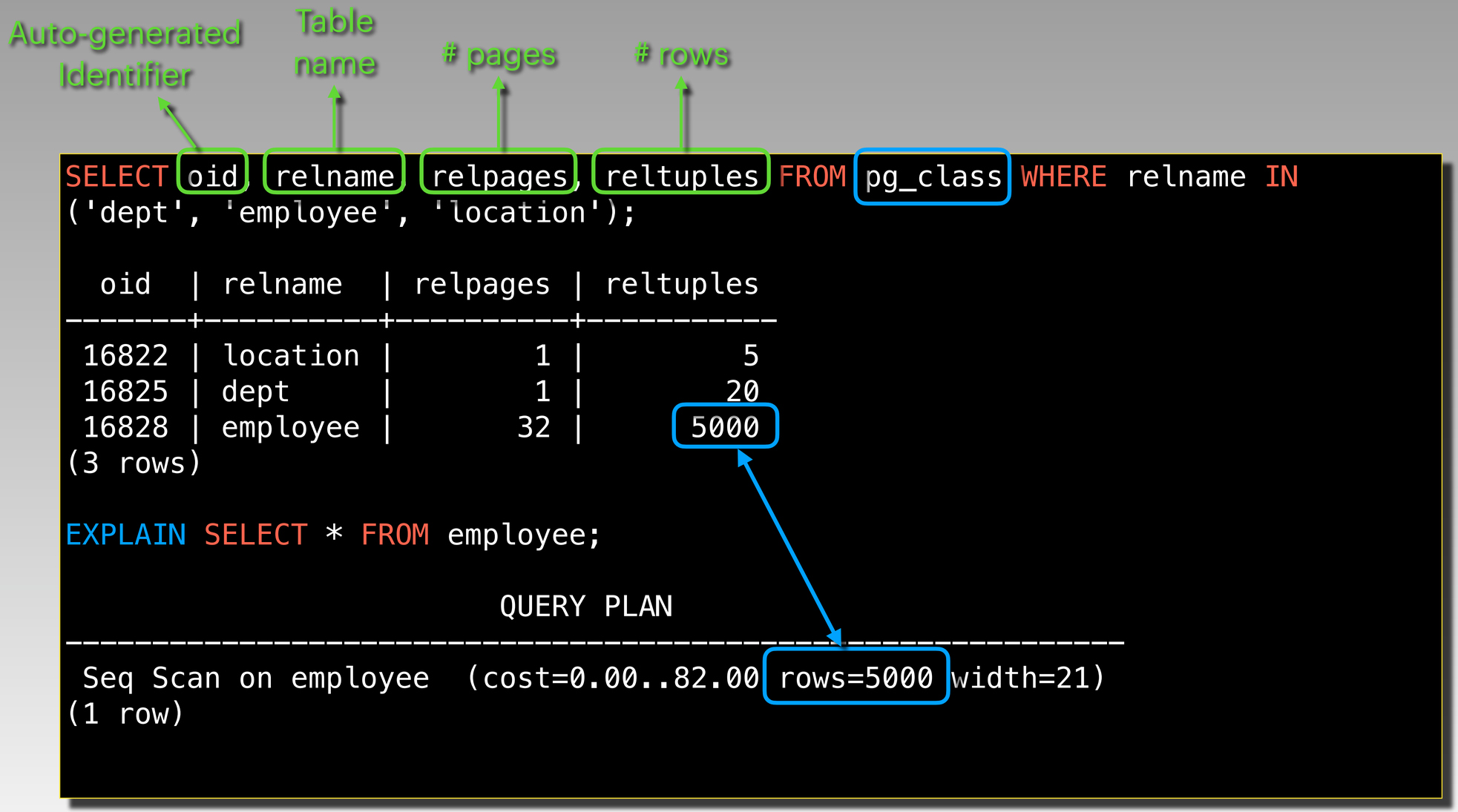
```
EXPLAIN SELECT e.id, e.name FROM employee e JOIN dept d ON e.dept_id = d.id
WHERE d.name='Engineering' ORDER BY e.name;

                                  QUERY PLAN
------------------------------------------------------------------------------
 Sort  (cost=114.47..115.09 rows=250 width=17)
   Sort Key: e.name
   ->  Hash Join  (cost=1.26..104.51 rows=250 width=17)
         Hash Cond: (e.dept_id = d.id)
         ->  Seq Scan on employee e  (cost=0.00..82.00 rows=5000 width=21)
         ->  Hash  (cost=1.25..1.25 rows=1 width=4)
               ->  Seq Scan on dept d  (cost=0.00..1.25 rows=1 width=4)
                     Filter: ((name)::text = 'Engineering'::text)
(8 rows)
```



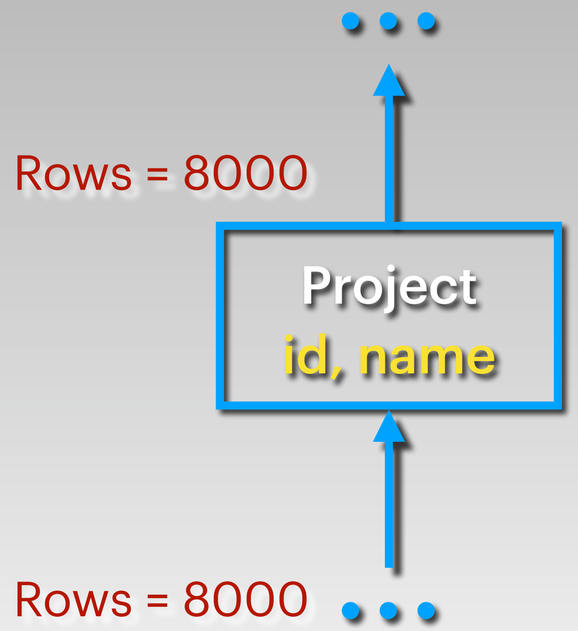
Leaf Nodes

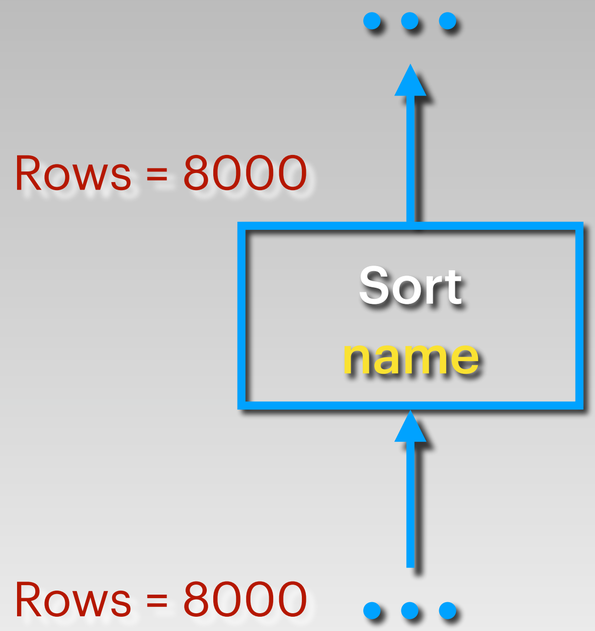




Non-leaf Nodes

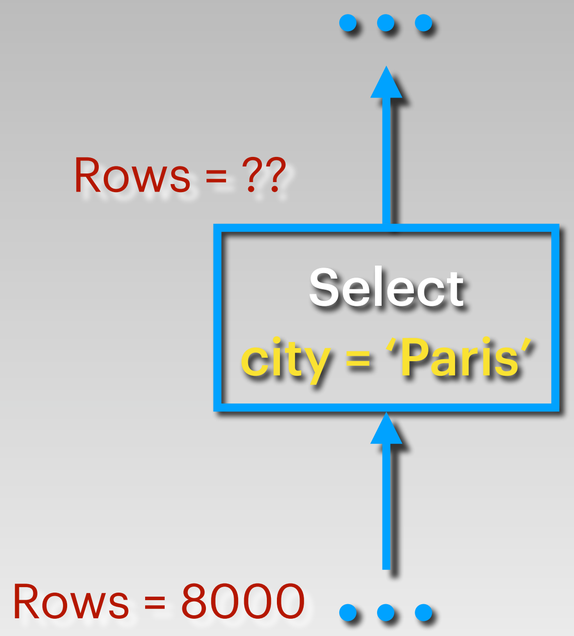
(1) Nodes with no cardinality Effect

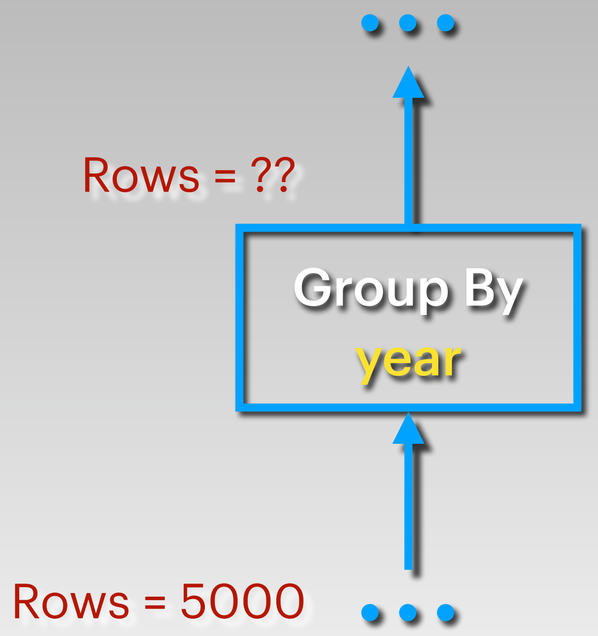


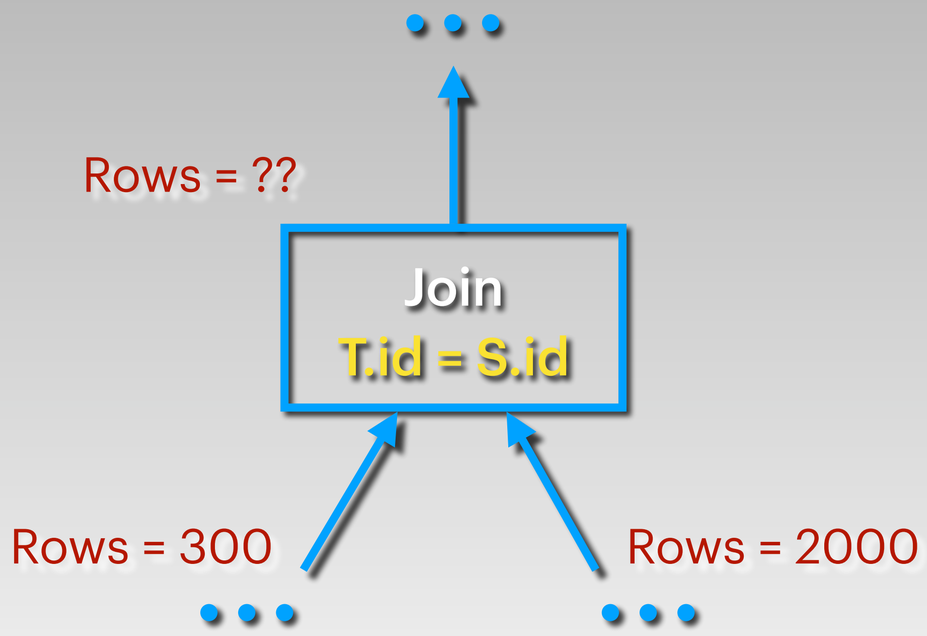


Non-leaf Nodes

(2) Nodes that change cardinality







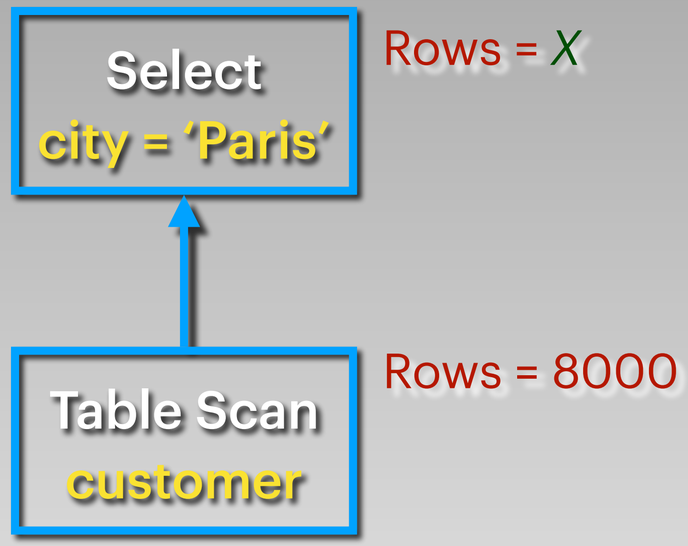
```
SELECT oid, relname, relpages, reltuples FROM pg_class WHERE relname IN
('dept', 'employee', 'location');

  oid  | relname  | relpages | reltuples
-------+----------+----------+-----------
 16822 | location |        1 |         5
 16825 | dept     |        1 |        20
 16828 | employee |       32 |      5000
(3 rows)

EXPLAIN SELECT e.id, e.name FROM employee e JOIN dept d ON e.dept_id = d.id
WHERE d.name='Engineering' ORDER BY e.name;

                                QUERY PLAN
-------------------------------------------------------------------------
 Sort  (cost=114.47..115.09 rows=250 width=17)
   Sort Key: e.name
   ->  Hash Join  (cost=1.26..104.51 rows=250 width=17)
         Hash Cond: (e.dept_id = d.id)
         ->  Seq Scan on employee e  (cost=0.00..82.00 rows=5000 width=21)
         ->  Hash  (cost=1.25..1.25 rows=1 width=4)
               ->  Seq Scan on dept d  (cost=0.00..1.25 rows=1 width=4)
                     Filter: ((name)::text = 'Engineering'::text)
(8 rows)
```

```sql
SELECT * FROM customer WHERE city = 'Paris';
```



X = 8000 * (ratio of rows that satisfy the filter)

selectivity ∈ [0, 1]

For a selection with filter *f*:

$$rows_{out} = rows_{in} * sel_f$$

Rows in ‘customer’ table = 8000

Value	Selectivity
Paris	0.04
London	0.06
New York	0.12
:	:

Sum = 1

```
SELECT * FROM customer WHERE city = 'London';
```

Rows = 8000 * 0.06 = *480*

Rows in ‘customer’ table = 8000

Value	Selectivity
Paris	0.04
London	0.06
New York	0.12
:	:

Sum = 1

```
SELECT * FROM customer WHERE city IN ('London', 'Paris');
```

Rows = 8000 * (0.06 + 0.04) = *800*

Rows in ‘customer’ table = 8000

No. of distinct values in ‘city’ column = 100

```
SELECT * FROM customer WHERE city = 'Chicago';
```

Rows = 8000 * 1/100 = *80*

Assumptions

- Inclusion
- Uniformity

Rows in ‘customer’ table = 8000

No. of distinct values in ‘city’ column = 100

```
SELECT * FROM customer WHERE city IN ('London', 'Paris');
```

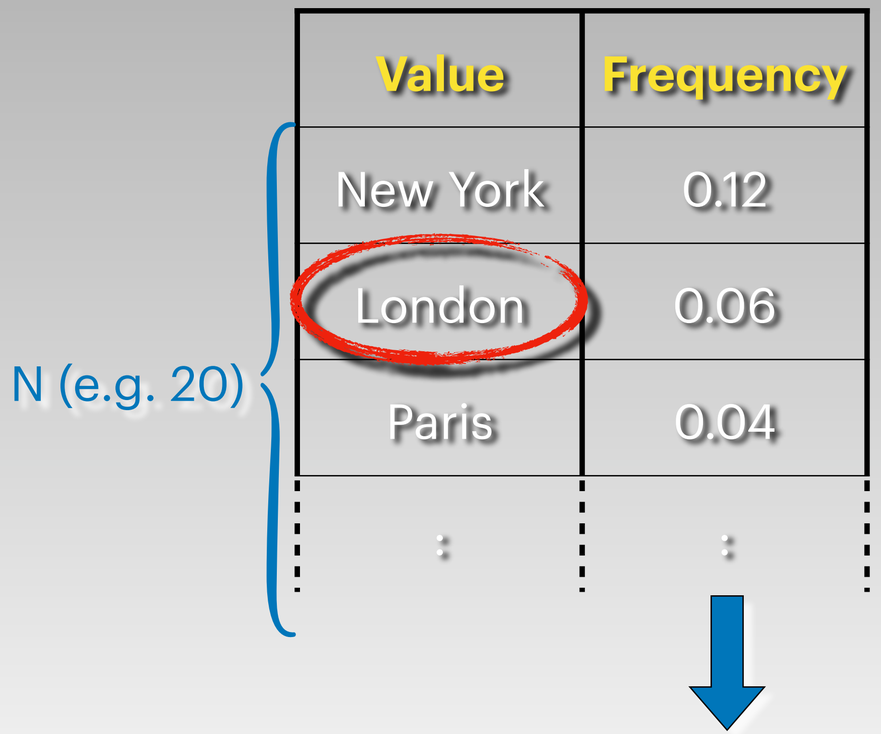
Rows = 8000 * 2/100 = *160*

Assumptions

- Inclusion
- Uniformity

Rows in ‘customer’ table = 8000

No. of distinct values in ‘city’ column = 100




```
SELECT * FROM customer WHERE city = 'London';
```

Rows = 8000 * 0.06 = *480*

Amr Elhelw's TECH VAULT

Rows in ‘customer’ table = 8000

No. of distinct values in ‘city’ column = 100

Value	Frequency
New York	0.12
London	0.06
Paris	0.04
:	:

N (e.g. 20)

Sum ≤ 1

(e.g. 0.98)

```
SELECT * FROM customer WHERE city = 'Chicago';
```

Rows = 8000 * 0.02/80 = *2*

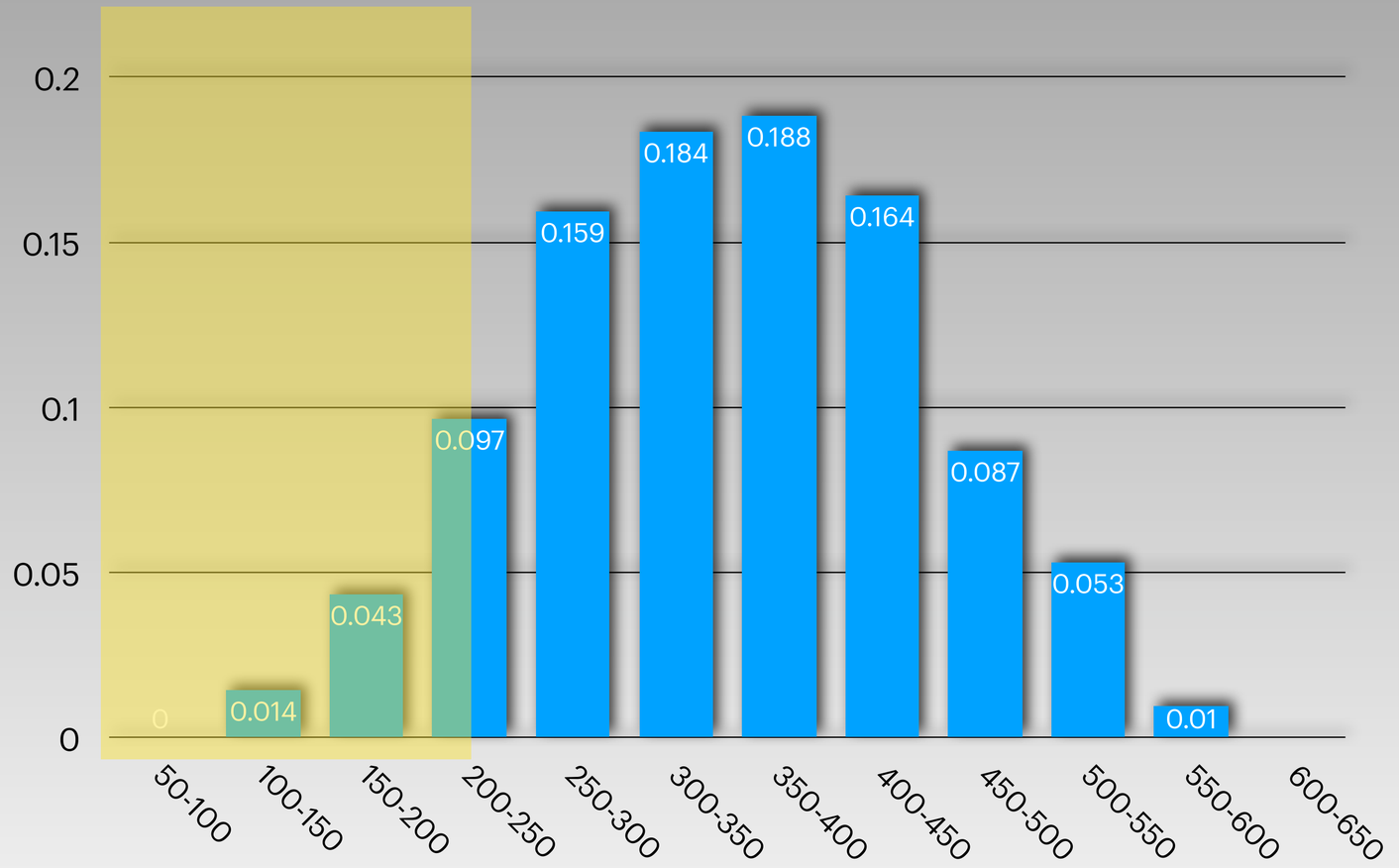
Assumptions for remaining values

- Inclusion
- Uniformity

- No. Of remaining values = 100 - 20 = 80
- Remaining frequency = 1 - 0.98 = 0.02
- Frequency of each remaining value = 0.02/80

Amr Elhelw's TECH VAULT

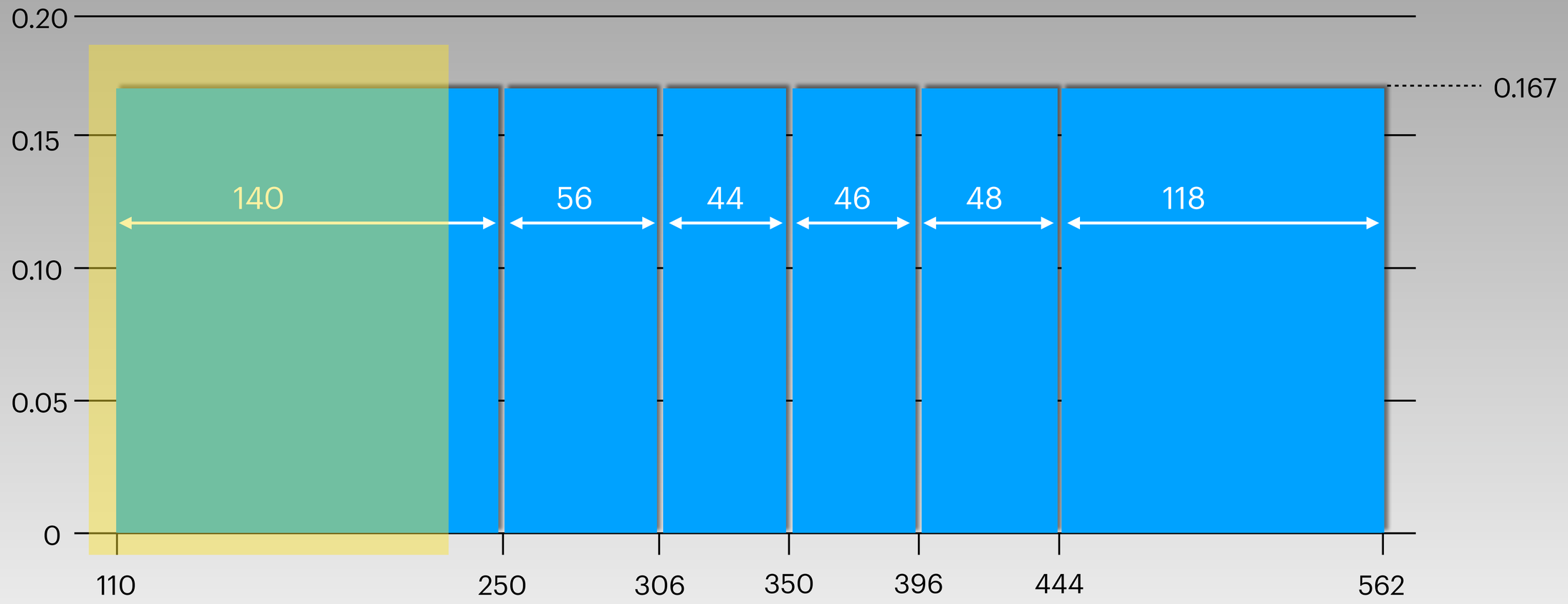
```
SELECT * FROM product WHERE price < 225;
```



Selectivity = 0.014 + 0.043 + (0.097/2) = 0.1055

Amr Elhelw's TECH VAULT

```
SELECT * FROM product WHERE price < 225;
```



Selectivity = 0.167 * (225 - 110)/(250-110) = 0.137

```
\d pg_stats;

                    View "pg_catalog.pg_stats"
        Column          |   Type    | Collation | Nullable | Default
------------------------+-----------+-----------+----------+---------
 schemaname             | name      |           |          |
 tablename              | name      |           |          |
 attname                | name      |           |          |
 inherited              | boolean   |           |          |
 null_frac              | real      |           |          |
 avg_width              | integer   |           |          |
 n_distinct             | real      |           |          |
 most_common_vals       | anyarray  |           |          |
 most_common_freqs      | real[]    |           |          |
 histogram_bounds       | anyarray  |           |          |
 correlation            | real      |           |          |
 most_common_elems      | anyarray  |           |          |
 most_common_elem_freqs | real[]    |           |          |
 elem_count_histogram   | real[]    |           |          |
```

```
SELECT * FROM product WHERE price < 300 AND category = ‘kitchen’;
```

Selectivity = 0.2

Selectivity = 0.1

Combined Selectivity = 0.2 * 0.1 = 0.02

Assuming Independence