ALUMNI

BY MTHREE CONSULTING

MTHREE ALUMNI DATA TRAINING

Data - Spreadsheet

instrument.csv #open in excel

- Auto expand columns

marketData.xlsx

- Freeze pane
- Bold headers
- View as table
- Row Filter
- Hide Columns
- Sort

marketDataNormalised.xlsx

- Normalisation
- Look at the various tabs
- Talk about file size

Data - Beyond spreadsheets

Excel can operate in 2 modes:

1. local files such as .csv, .txt, .xlsx
2. ODBC/OLE DB

Local files become unmanageable when:

- Data becomes large(65K+), SS stops working/becomes really slow
- User interface becomes clunky, not programmatic. VBA is too complex for casual users.
- A SS is basically a database underneath, real databases can have multiple viewers including:
  - Excel
  - Web
  - SQLDeveloper
  - etc
- DB allows 1 centralised copy with multiple user access

SQL

SQL Reading List

- http://www.w3schools.com/sql/
- https://www.mysql.com/
- https://www.codecademy.com/
- https://www.techonthenet.com/oracle/index.php

SELECT Statement

```
SELECT columns FROM table;
```

Syntax of a select query

```
select * from posttrade.orders;
```

Retrieve all columns, all rows from the orders table

```
sEleCt * fROm posttrade.ordERS;
```

case insensitivity

```
select datetime from posttrade.orders;
```

Retrieve one column

```
select datetime,price from posttrade.orders;
```

Get 2 columns

```
select datetime,price,price from posttrade.orders;
```

Get 3 columns, one of which is duplicated

```
select side,posttrade.orders.* from posttrade.orders;
```

Output Column - Names

```
select datetime, instrument, side from
posttrade.orders;
```

A more interesting set of columns

```
select datetime as OrderTime, instrument as
"Symbol", side "BuySell" from posttrade.orders;
```

In the output, rename datetime column to be called TradeTime, rename instrument to Symbol and side to BuySell

Effect of "

Distinct

```
select distinct side from posttrade.orders;
```

How many sides are there?

```
select distinct ordstatus from posttrade.orders;
```

How many stages can an order be in?

```
select distinct side,ordstatus from
posttrade.orders;
```

Show unique combinations of side and ordstatus that are present in the table

Joining Strings

```
select tag,value,tag||'='||value from
refdata.fix;
```

Show every FIX tag+value, and construct all possible fixtags

```
select tag,value,tag||'='||value fixtag from
refdata.fix;
```

Name the column better

Arithmetic

```
select price,orderqty,price*orderqty
dollarvalue from posttrade.orders;
```

Get the price, orderqty columns from the orders table. Also generate a third column named dollarvalue with the values of the price column multiplied by the orderqty column

```
select price,orderqty,cumqty,price*cumqty
dollarvalue,cumqty/orderqty fillratio from
posttrade.orders;
```

Add another column showing how close the order is to being fully filled

```
select 2,3,2*3,2*3+4,datetime from
posttrade.orders;
```

PEDMAS/BODMAS. Generate 4 columns which contain 4 unique values, and select the datetime column from orders

```
select 2,3,2*3,2*3+4,4+3*2 from dual;
```

Every query must return a table, so there is a special table for checking maths, and current_timestamp, etc, called dual

Numbers

- 123 vs 1111011

Whole numbers

type	size(bytes)	min	max	default
byte	1	-128	127	0
short	2	-32768	32767	0
int	4	-2^{31}	$2^{31}-1$	0
long	8	-2^{63}	$2^{63}-1$	0L

Floating point numbers

varchar2, char, date

type	size(bytes)	default
float	4	0.0f
double	8	0.0d

Data Types

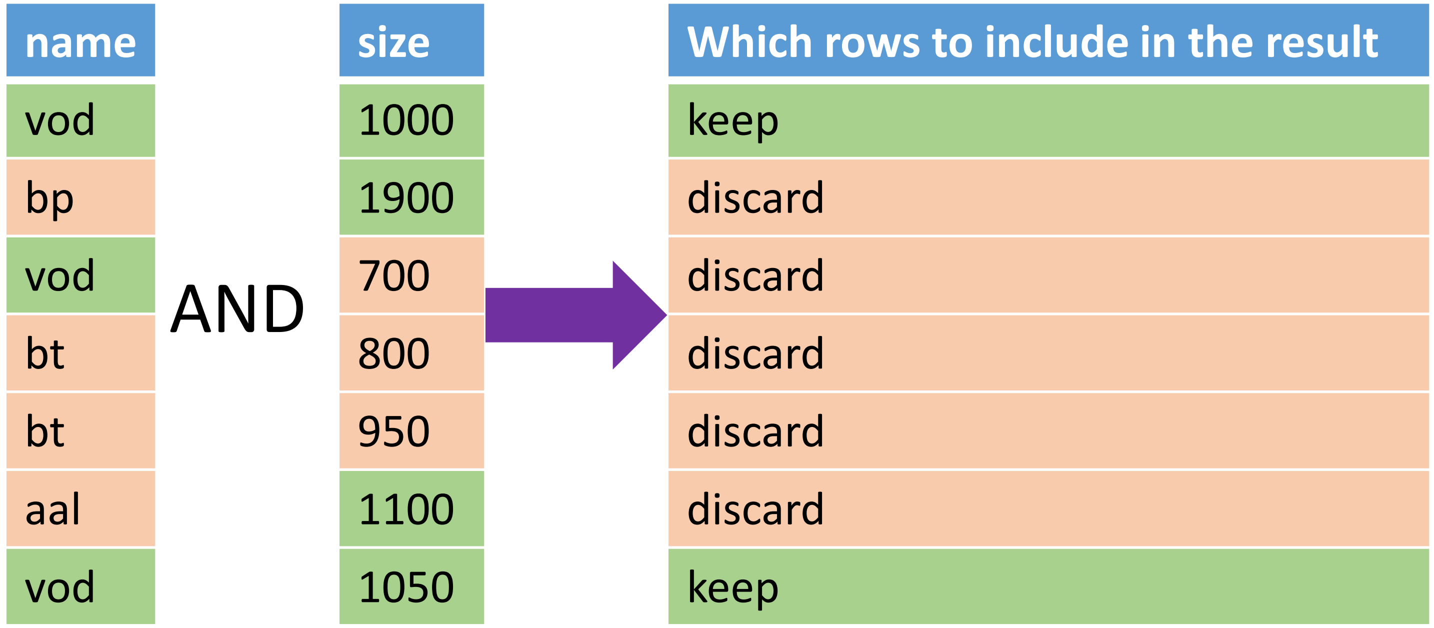
Data Types	Description	Example (Data Type/Value)
NUMBER	• Fixed and Floating point numbers	*NUMBER (38,0) :* **17**
VARCHAR2	• Variable-length character string • Holds letter and numbers	*VARCHAR2 (100 BYTE):* **Send 2 remaining chunks to market**
CHAR	• Fixed-length character string	*CHAR (4 BYTE):* **XLON**
FLOAT	• 'Floating point number' – no fixed number of digits before or after decimal point • Subtype of Number	*FLOAT (126):* ***540.76***
TIMESTAMP	• Date and Time • Year, Month, Day, Hour, Minute, Second	*TIMESTAMP (6):* **18-Jul-16 07.55.00.000000000**

Row Filter

```
SELECT columns FROM table WHERE constraints;
```

Syntax of WHERE

```
select * from fakeTable where name='vod' AND size>=1000
```



name	size	Which rows to include in the result
vod	1000	keep
bp	1900	discard
vod	700	discard
bt	800	discard
bt	950	discard
aal	1100	discard
vod	1050	keep

```
select instrument,orderqty from posttrade.orders where side=1;
```

Show all the buy orders

```
select side,instrument,orderqty from posttrade.orders where side<>1;
```

Show the rest, i.e. sells

```
select instrument,orderqty,cumqty from posttrade.orders where
cumqty>0;
```

Show orders that are at least partially filled

```
select datetime,instrument,orderqty from posttrade.orders where
datetime<='18-JUL-16 08:00:00.000000000';
```

Show all the orders before 8am on 18th July 2016

LET’S TAKE A SHORT BREAK

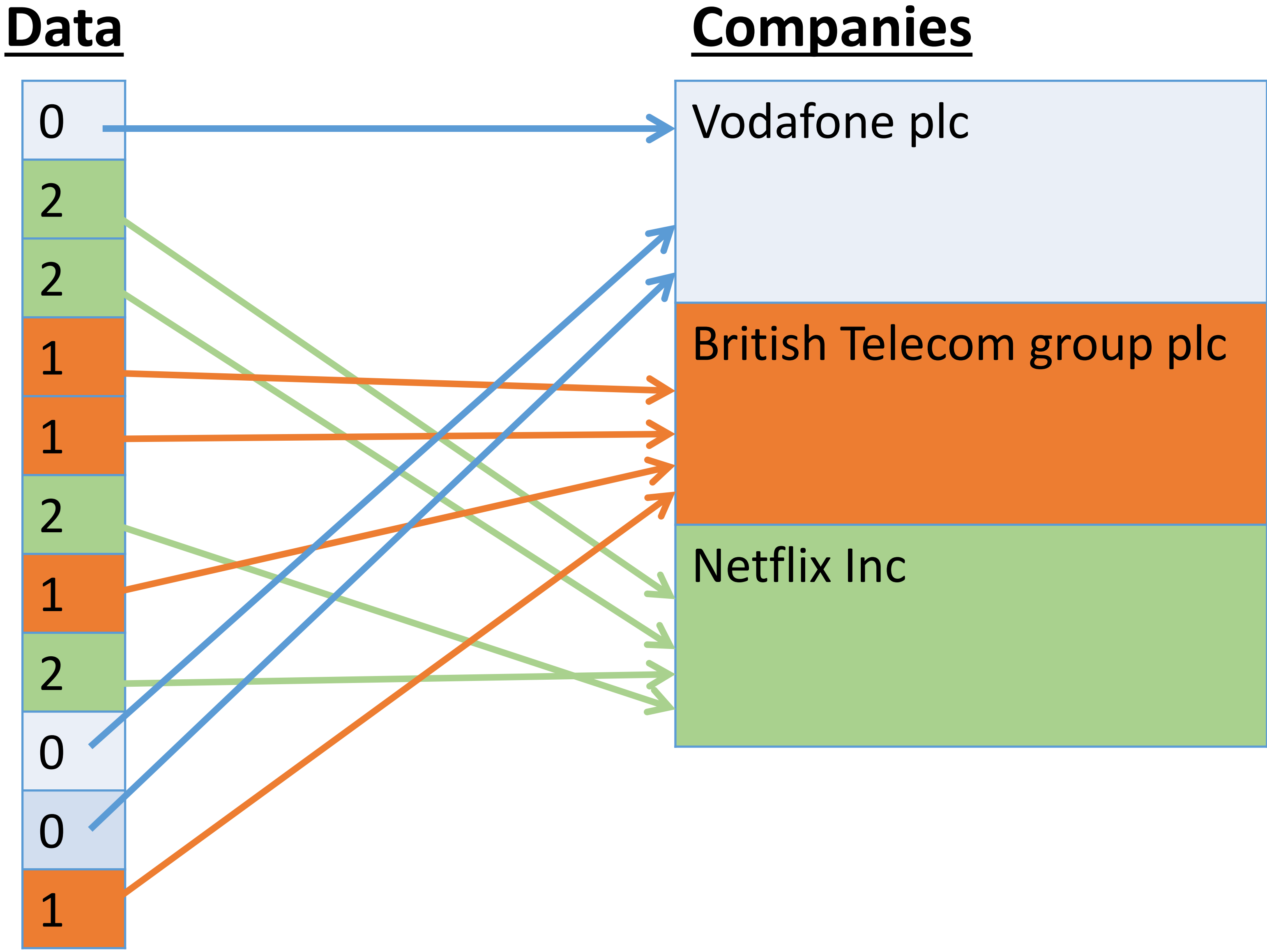
Data Width

- Fixed width columns:
  - allow faster data retrieval especially in column stores.
  - element x in a list of 4 byte elements can be found at position 4x

- Variable width columns
  - scan the list, use terminators after values **'dave\000hodgins\000'**
  - create an index recording the starting position of each element **'0 5' 'dave\000hodgins\000'**

Enumeration

- Enumeration:- Replacing a list of duplicated values with their position in a unique list
- Space
- Speed
- Ease of change



Data

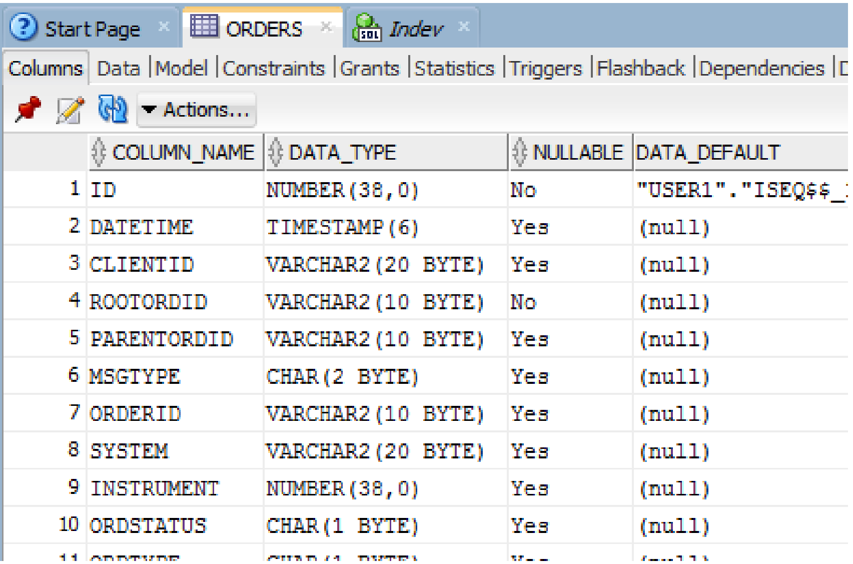
0
2
2
1
1
2
1
2
0
0
1

Companies

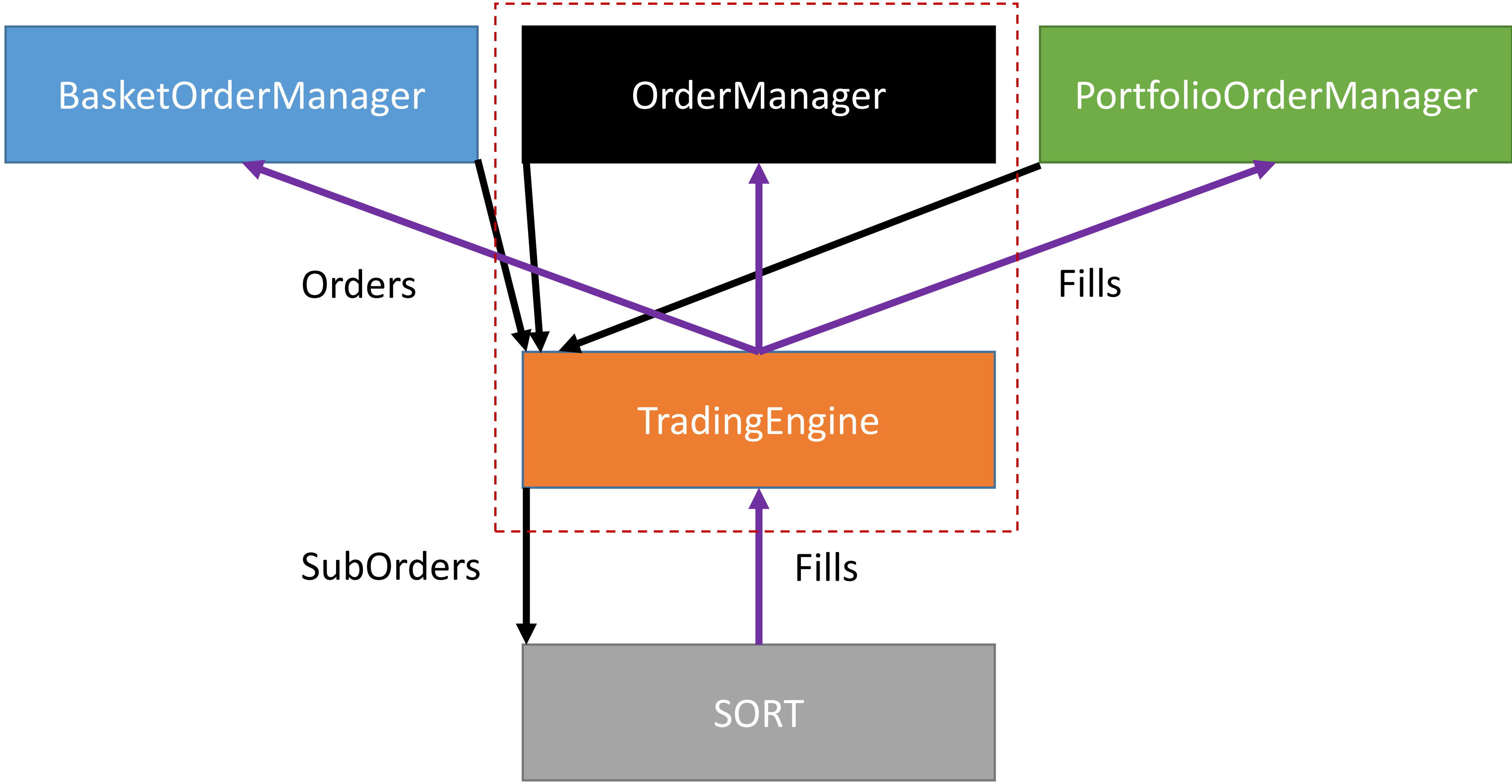
Vodafone plc
British Telecom group plc
Netflix Inc

Order Messages

describe posttrade.orders;


Start Page
ORDERS
Indev
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | D
Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT
1	ID	NUMBER(38,0)	No	"USER1"."ISEQ$$_
2	DATETIME	TIMESTAMP(6)	Yes	(null)
3	CLIENTID	VARCHAR2(20 BYTE)	Yes	(null)
4	ROOTORDID	VARCHAR2(10 BYTE)	No	(null)
5	PARENTORDID	VARCHAR2(10 BYTE)	Yes	(null)
6	MSGTYPE	CHAR(2 BYTE)	Yes	(null)
7	ORDERID	VARCHAR2(10 BYTE)	Yes	(null)
8	SYSTEM	VARCHAR2(20 BYTE)	Yes	(null)
9	INSTRUMENT	NUMBER(38,0)	Yes	(null)
10	ORDSTATUS	CHAR(1 BYTE)	Yes	(null)
11	ORDTYPE	CHAR(1 BYTE)	Yes	(null)



Constraints

```
select
rootordid,parentordid,msgtype,orderid,system,ordstatus,orderqty,cumqty,leavesqty,price,descr from posttrade.orders
where rootordid='om1';
```

Show all the updates and children for a single order

```
select clientid,datetime,instrument,orderqty,side from
posttrade.orders where parentordid is null;
```

Show only the first version of each order, before it gets updated

```
select datetime,instrument,orderqty,side, parentordid from
posttrade.orders where parentordid like 'te%';
```

Show all the orders that were created by, or a child of an order created by the TradingEngine

```
select datetime,parentordid,instrument,orderqty,side from
posttrade.orders where parentordid like 'TE%';
```

SQL is not case sensitive, but like and = are sensitive, there are no parentordid's that start with 'TE' followed by any characters

```
select msgtype,instrument,orderqty,side from
posttrade.orders where msgtype in ('UP','UC');
```

Show only parent and child orders, orders start their life without a msgtype, so this will only show their children and grandchildren

```
select ordstatus,instrument,orderqty,side from
posttrade.orders where ordstatus not in (1,2);
```

Show rows which don't have ordstatus=1 and don't have ordstatus=2

```
select id,msgtype,instrument,orderqty,side from
posttrade.orders where id between 80 and 85;
```

id is a column we are using to number the records, show only records 30,31,32,33,34,35

Constraints(cont.)

```
select id,msgtype,instrument,orderqty,side from
posttrade.orders where id not between 70 and 75;
```

Get all rows except the ones we saw earlier

```
select datetime,instrument,orderqty,side from
posttrade.orders where datetime like '% 07.%';
```

Show rows generated at 7AM any day

```
select * from refdata.instrument where name like 'B_ %';
```

Show all the companies that start with a B followed by any single character followed by a space, followed by any characters

```
select * from refdata.instrument where name like 'B_ %' and
mic is not null;
```

All the companies with 2 letter words at the start of their name, starting with B, which have an exchange

```
select * from refdata.fix where regexp_like(descr,'Re[pj]');
select * from refdata.fix where
regexp_like(descr,'Rep.*Rep');
```

PCRE version of like

```
select datetime,price,orderqty,cumqty from posttrade.orders
where cumqty is not null and side='1';
```

All the buys which are partly/fully filled

```
select * from posttrade.orders where datetime between '18-
JUL-16 08.00' and '18-JUL-16 08.00.00.02';
```

Time comparisons automatically add 0's at the end

Operator Precedence

Operator	Examples	Type
1	+ / - *	Arithmetic operators
2	||	Concatenation operator
3	= < > <= >=	Comparison conditions
4	is null, is not null, like, in, not in	
5	between, not between	
6	<> !=	
7	not	
8	and	
9	or	

```
select * from refdata.instrument where instrument is not
null or ric is not null;
```

Get rows where there is a non null value in either instrument of ric columns

```
select * from refdata.instrument where mic='XLON' or MIC is
null and isin is null;
```

Show london symbols and rows with no exchange and no isin

```
select * from refdata.instrument where mic='XLON' or (MIC
is null and isin is null);
```

Same as above

```
select * from refdata.instrument where (mic='XLON' or MIC
is null) and isin is null;
```

Show rows with have no isin, and either have no MIC or are london instruments

Exercise

1. Which fixtags have i recorded?
2. Repeat 1 but name the result column ***fixtags***
3. Using the ***FIX*** table as a manual reference, take the relevant columns from the ***ORDERS*** table and construct a series of FIX messages

4. Show the rows in ***ORDERS*** where ***LEAVESQTY*** is equal to the difference between ***ORDERQTY*** and ***CUMQTY***
5. Which rows in ***ORDERS*** have ***LEAVESQTY*** not equal to the difference between ***ORDERQTY*** and ***CUMQTY***?
6. Calculate the difference between Pi(to within 5d.p.) and 22/7
7. Retrieve all bob's orders, but only show completed or original order rows.
8. Show all the rows in ***FIX*** where the ***descr*** column mentions the word ***'short'***.

Sorting

Functions

Joins

Aggregation

Creating/changing data

Quiz: What order will the rows be in, and why?

```
select * from posttrade.orders;
```

Sorting

```
select * from posttrade.orders order by side;
```

Get the whole orders table sort the rows by the side column in ascending order

```
select * from posttrade.orders where cumqty is not null order by side;
```

Only partially/fully filled orders, buys first, sells last

```
select * from posttrade.orders where cumqty is not null order by side desc;
```

Sells first, buys last

```
select * from refdata.fix order by tag desc,value asc;
```

First sort by value in ascending order, then sort by tag in descending order

```
select * from refdata.instrument order by instrument desc;
```

Nulls come at the start

```
select * from refdata.instrument order by instrument;
```

Nulls come at the end

Functions

`select id,tag,name,value,upper(descr),descr from refdata.fix;`	
`select id,tag,name,value,descr,upper(descr) from refdata.fix where upper(descr)='BUY';`	**Include both the original descr and the uppercase version**

`select ric,instr(ric,'.') from refdata.instrument;`
`select ric,instr(ric,'.'),substr(ric,3) from refdata.instrument;`
`select ric,instr(ric,'.'),substr(ric,instr(ric,'.')) from refdata.instrument;`
`select ric,instr(ric,'.'),substr(ric,instr(ric,'.')) mycolumn1 from refdata.instrument;`
`select ric, 1+instr(ric,'.'),instr(ric,'.'),substr(ric,instr(ric,'.')),substr(ric,1+instr(ric,'.')) mycolumn1 from refdata.instrument;`	

	instr searches each value for the character and returns the position of the first occurrence. substr chops up strings from the position specified until the end or for the specified number of characters

`select isin,sedol,substr(isin,5,7)shortisin from refdata.instrument where substr(isin,5,7)<>sedol;`
`select isin,sedol,substr(isin,5,7)shortisin from refdata.instrument where substr(isin,5,7)=sedol;`
	Get characters 5-12 from each isin and show rows where this was the same as the sedol

Exercise

1. Using the **INSTRUMENT** table, calculate which instruments contain the **INSTRUMENT** column within the BBID column.

2. What are the exceptions (to question 1)?

3. Normally the formula to construct a BBID is:

INSTRUMENT:EXCHANGE

eg: BT is the instrument
LN is the exchange
BT:LN is the BBID

Show me when, the formula not INSTRUMENT:Something by writing some SQL ?

Functions(cont.)

```
select rootordid,replace(rootordid,'om','OM_')from
posttrade.orders;
```

Make rootordid format nicer

```
select
cumqty,orderqty,100*cumqty/orderqty,floor(100*cumqty/orderqty)
,ceil(100*cumqty/orderqty)from posttrade.orders where cumqty
is not null;
```

Show percentage filled rounded down

```
select orderqty,mod(orderqty,40) from posttrade.orders;
```

Show number of whole lots column and which orders are not multiples of whole lots.

```
select 1,abs(-
42),abs(42),exp(1),log(2,32),round(log(2,32)),power(10,6) from
dual;
```

abs returns a positive number, exp is e

Type Conversion

```
select datetime,add_months(datetime,6),next_day(datetime,'fri')
from posttrade.orders;
```

Add 6 months to each date, determine the date of the following Friday for each date

```
select sysdate,months_between(sysdate,'01-JAN-2000') from dual;
```

Calculate how many months are between 1st Jan 2000 and now

```
select 100*cumqty/orderqty||'%' from posttrade.orders;
```

Implicit conversion of number to characters

```
select 1+value from refdata.fix;
select value,1+value from refdata.fix where value<'A';
```

More implicit conversions

```
select datetime,
 to_char(datetime,'YYYY.MM.DD'),
 to_char(datetime,'HH24:MI:SS.FF9'),
 to_date(to_char(datetime,'DD.Mon.YY')),
 to_timestamp(to_date(to_char(datetime,'DD.Mon.YY')))
from posttrade.orders;
```

Conditionals

```
select instrument,ric,isin,
coalesce(instrument,ric,isin)identifier
from refdata.instrument;
```

Create identifier column with values from instrument unless null then ric unless also null then isin

```
select orderqty,cumqty,ordstatus,
 case ordstatus
  when '0' then 'new'
  when '1' then 'partially filled'
  when '2' then 'fully filled'
 end
from posttrade.orders;
```

Choose different values for the column based on the values in ordstatus

```
select orderqty,cumqty,ordstatus,
 case when cumqty=0 or cumqty is null then 'new'
      when cumqty=orderqty then 'fully filled'
  else 'partially filled'
 end
from posttrade.orders;
```

More complex case statement, not limited to the values of a single column

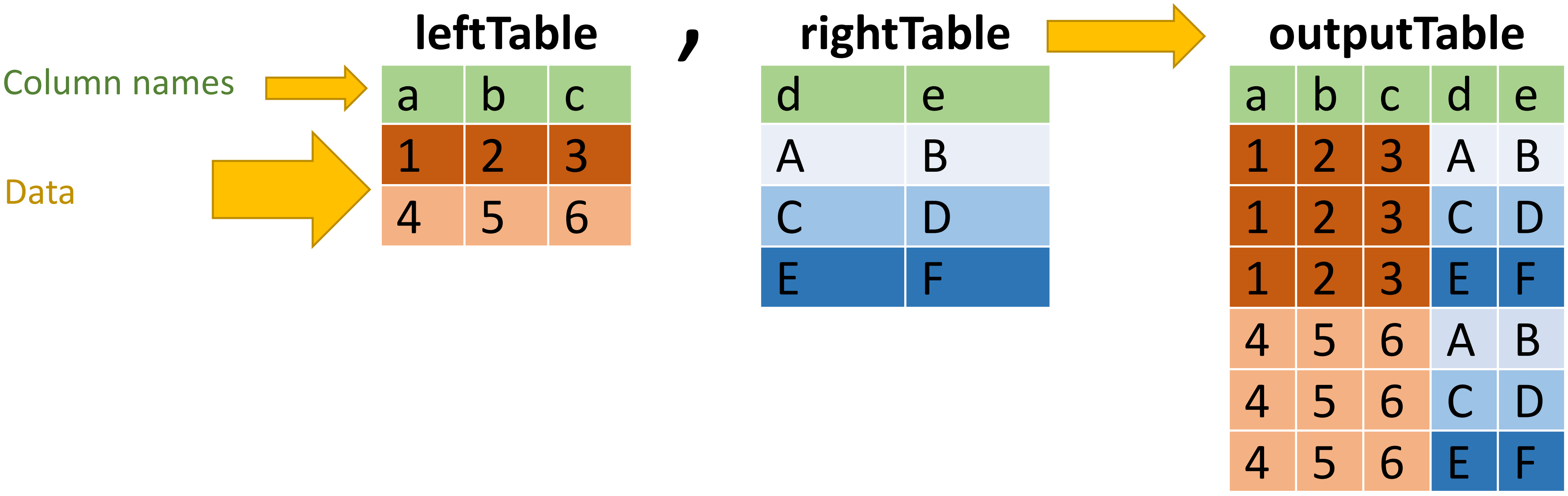
Exercise

1. Retrieve all the rows from **FIX** which have the word 'limit' in their description, your search should not be case sensitive
2. 12 character ISIN codes normally start with a 2 character country code, retrieve a unique list of country codes from the instrument table
3. Get the fills from **ORDERS**, show the value of each fill, show side as 'Buy' and 'Sell' instead of numbers, show the largest sell first.
    1. break this into steps
    2. check the steps with me
    3. implement the steps
4. Repeat(1) but use at least 4 different methods including:
    1. REGEXP_LIKE(DESCR, 'limit','i'); --where i is case insensitive(options other than 'i' are never used normally)

Joins

Cartesian Join

- What is a join?
- Cartesian join, all combinations of rows from each table



Column names

Data

leftTable ,

a	b	c
1	2	3
4	5	6

rightTable

d	e
A	B
C	D
E	F

outputTable

a	b	c	d	e
1	2	3	A	B
1	2	3	C	D
1	2	3	E	F
4	5	6	A	B
4	5	6	C	D
4	5	6	E	F

```
select * from posttrade.orders,refdata.instrument;
```

Cartesian join

```
select orders.id,datetime,orders.instrument,instrument.id,ric from
posttrade.orders,refdata.instrument;
```

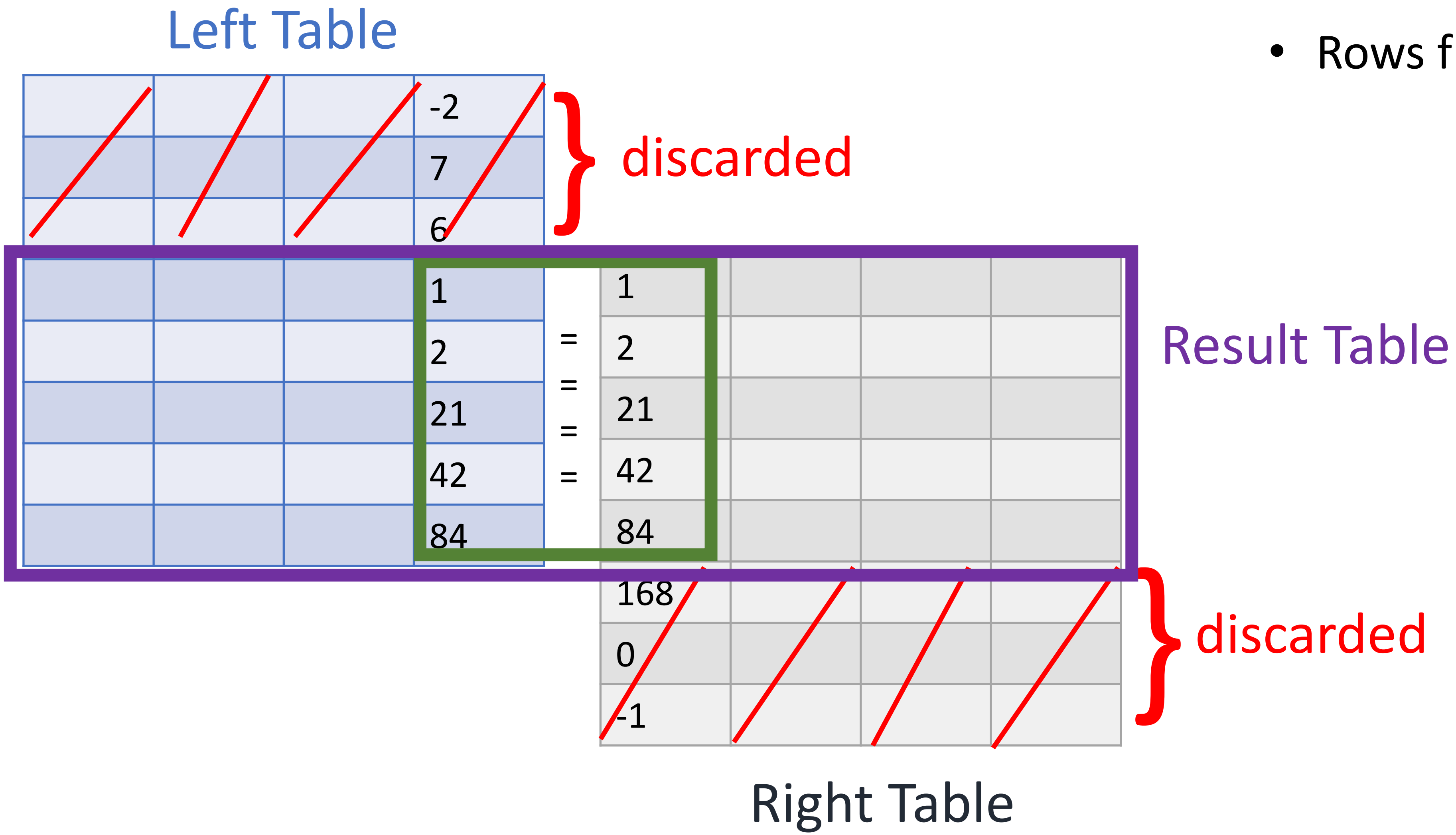
Select columns from each table, use . notation to disambiguate

```
select orders.id,datetime,orders.instrument,instrument.id,ric from
posttrade.orders,refdata.instrument where
orders.instrument=instrument.id;
```

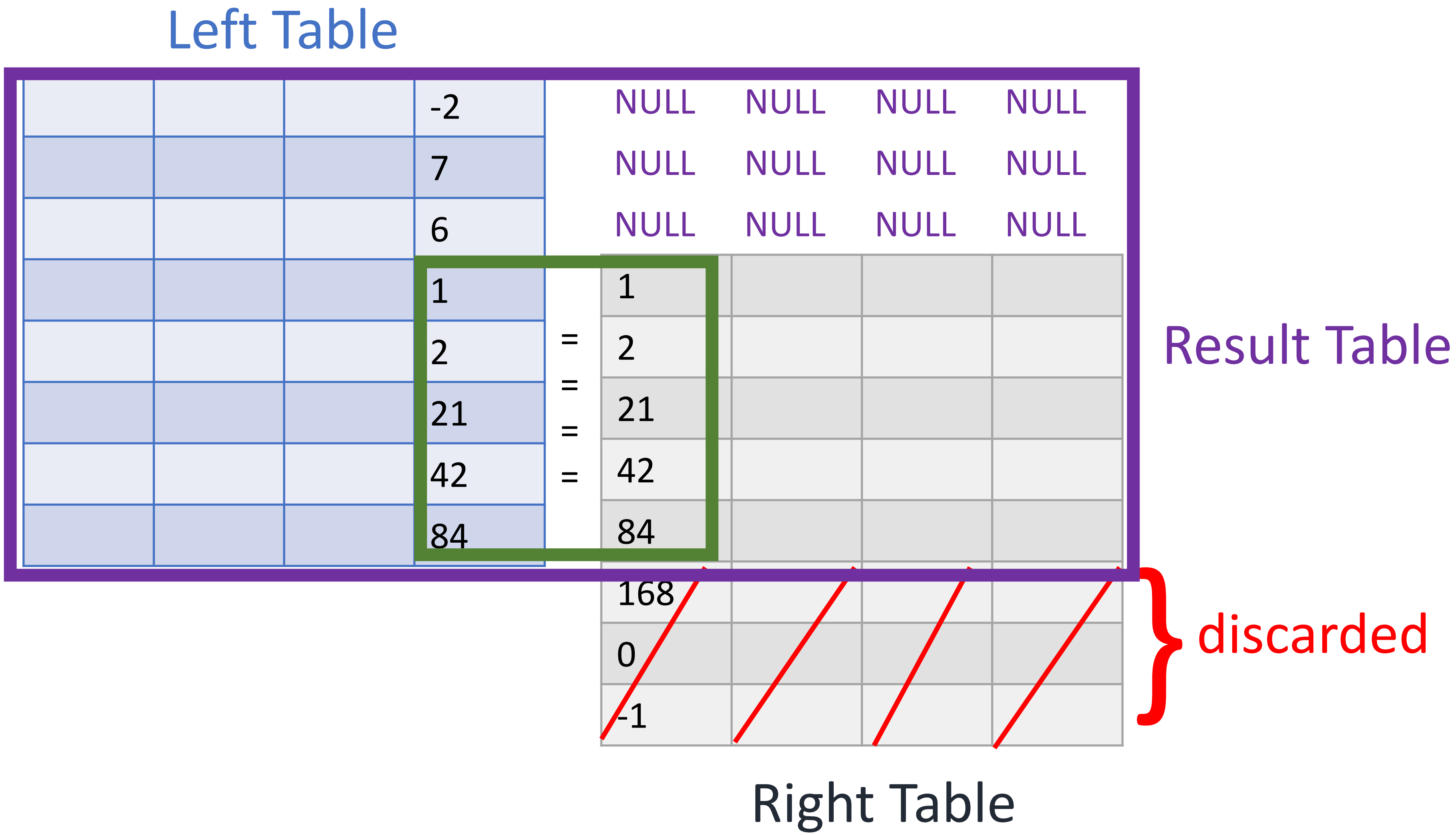
(lookup)Only show rows where the columns match

Inner Join

- Rows from both tables or neither



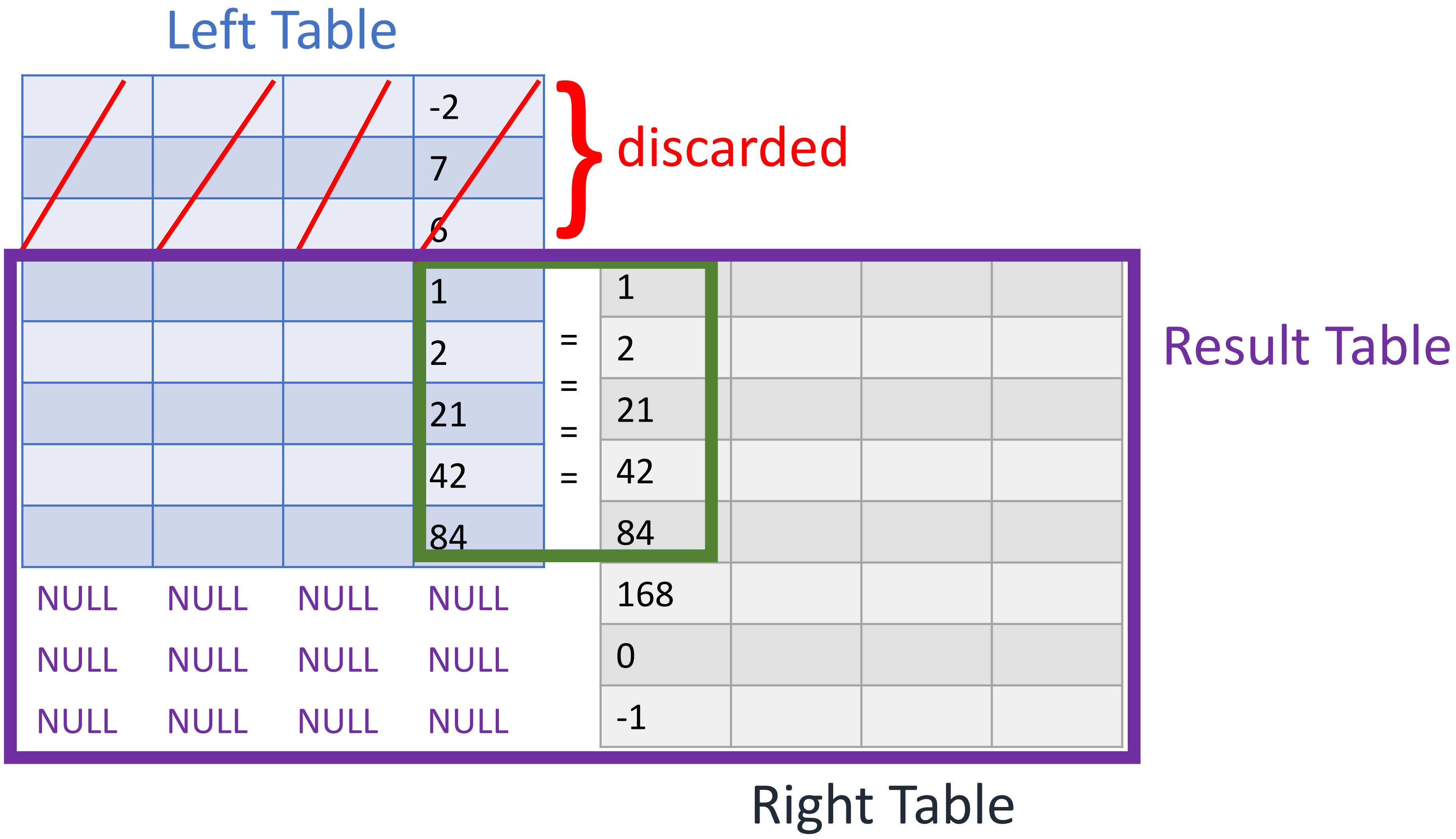
Left Outer Join



Outer Joins

- + Inner join
- + All of one table
- + nulls from the other

Right Outer Join



Left Table

discarded

Result Table

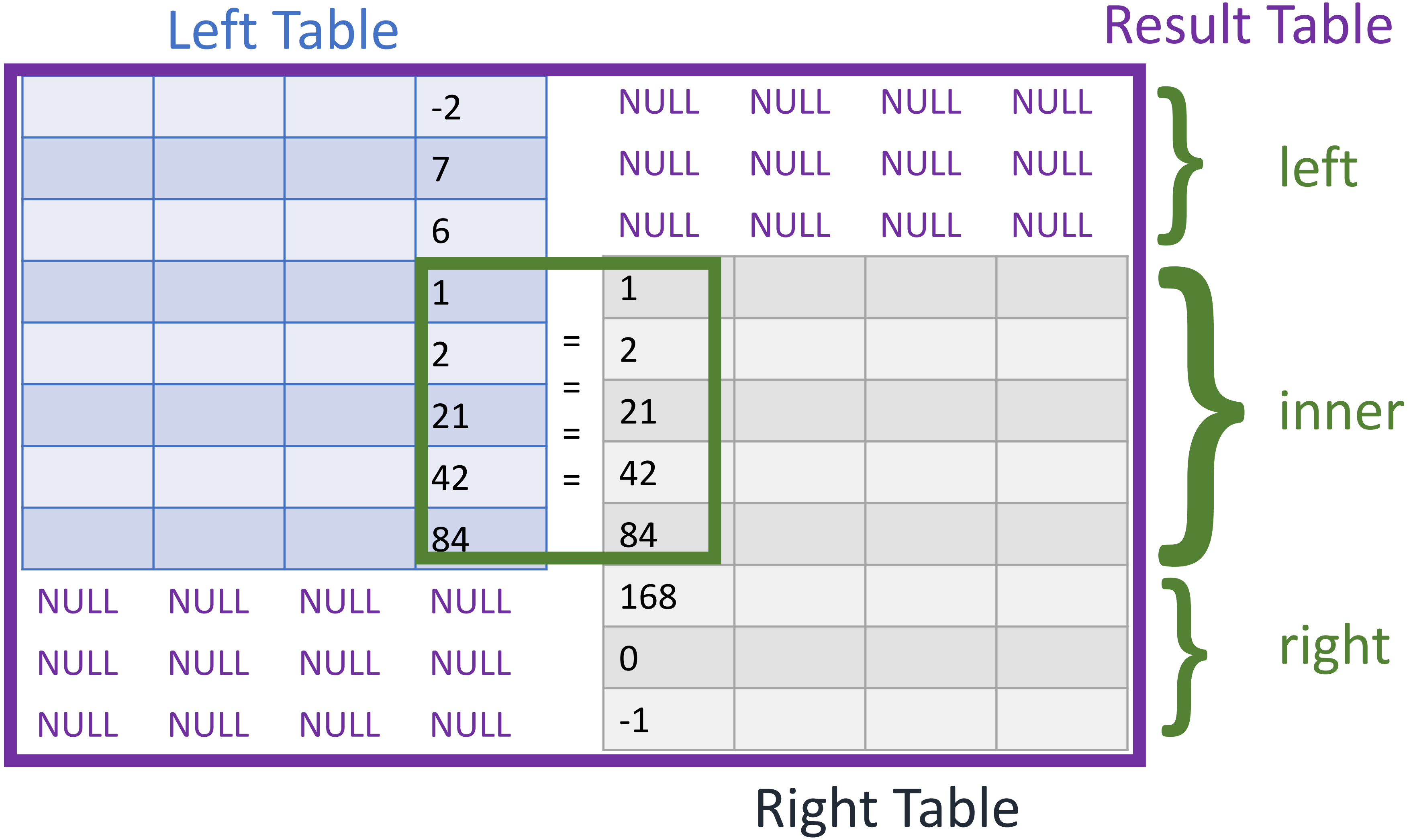
Right Table

			-2					
			7					
			6					
			1		1			
			2	=	2			
			21	=	21			
			42	=	42			
			84	=	84			
NULL	NULL	NULL	NULL		168			
NULL	NULL	NULL	NULL		0			
NULL	NULL	NULL	NULL		-1			

Outer Joins

- Inner join
- All of one table
- nulls from the other

Full Outer Join



Left Table

Result Table

			-2	NULL	NULL	NULL	NULL
			7	NULL	NULL	NULL	NULL
			6	NULL	NULL	NULL	NULL
			1	1			
			2	2			
			21	21			
			42	42			
			84	84			
NULL	NULL	NULL	NULL	168			
NULL	NULL	NULL	NULL	0			
NULL	NULL	NULL	NULL	-1			

Right Table

Full Outer Joins

- + All of both tables
- + Nulls

Left Join Theory

- Cartesian join then filter
- Logically all the rows from the left table/column enriched with the matching rows in the right table/column

leftTable

a	b	c
4	5	6
7	8	9
1	2	3

leftTable.a=rightTable.a

rightTable

a	e
1	B
2	D
4	F

outputTable

a	b	c	e
4	5	6	F
7	8	9	
1	2	3	B

Right Join Theory

- Logically all the rows from the right table/column enriched with the matching rows in the left table/column

leftTable

a	b	c
1	2	3
4	5	6
7	8	9

leftTable.a=rightTable.a

rightTable

a	e
1	B
2	D
4	F

outputTable

a	b	c	e
1	2	3	B
2			D
4	5	6	F

Left Join Examples

```
select * from posttrade.orders;
```

```
select * from posttrade.orders where ordstatus=0 order by
rootordid,datetime;
```

Show only the first version of orders exclude subsuquent updates

```
select orderid from posttrade.orders where ordstatus=0 order by
rootordid,datetime fetch first 1 rows only;
--and sysdate==to_date(datetime)
```

Get the first order

```
select * from posttrade.orders where rootordid=(select orderid from
posttrade.orders where ordstatus=0 order by rootordid,datetime fetch
first 1 rows only);
```

Get all it's children

```
select * from
(select id p_id,orderid p_orderid from posttrade.orders where
ordstatus=0),posttrade.orders
where parentordid=p_orderid(+) and rootordid='om1' order by id;
```

Left join - All rows from left hand column, add information from the right

```
select * from
posttrade.orders left outer join (select id p_id,orderid p_orderid from
posttrade.orders where ordstatus=0)
on parentordid=p_orderid
where rootordid='om1' order by id;
```

Modern syntax

Left Join Examples Cont.

`select * from refdata.product;`	Show the futures and options product data
`select * from refdata.contract;`	Show the futures and options contract data
`select * from refdata.contract left outer join refdata.product on refdata.contract.productCode=refdata.product.productCode;`	Enrich the contract data with the product data

Exercise(30 mins)

1. Spend some time trying out left joins.
2. Try my orders left join query without the `ordstatus=0`
3. determine the name of everyones manager, and then try joining the other tables:

```
    user1.emp
    user1.dept
    user1.salgrade
```

Right Join Example

```
select * from
(select id p_id,orderid p_orderid from posttrade.orders where
ordstatus=0),posttrade.orders
where p_orderid(+)=parentordid and rootordid='om1' order by id;
```

Right join - All rows from right hand column, add information from the left

```
select * from
(select id p_id,orderid p_orderid from posttrade.orders where
ordstatus=0) right outer join posttrade.orders
on parentordid=p_orderid
where rootordid='om1' order by id;
```

Modern syntax

Other Joins

```
select * from posttrade.orders,refdata.instrument where
posttrade.orders.instrument=refdata.instrument.id;
```

Inner join old

```
select * from posttrade.orders inner join refdata.instrument on
posttrade.orders.instrument=refdata.instrument.id;
```

Inner join new

```
select * from posttrade.orders,refdata.instrument,refdata.mic where
posttrade.orders.instrument=refdata.instrument.id and
refdata.instrument.mic=refdata.mic.mic;
```

3 way join

```
select case when m.mic is not null and i.mic is not null then 'both - inner'
    when m.mic is null then 'instrument - left'
    else 'mic - right'
  end help,
 ric,m.mic m_mic,i.mic i_mic,name,descr
from refdata.instrument i full outer join refdata.mic m
   on i.mic=m.mic
order by i_mic,m_mic;
```

Full outer join - Keep rows from both table, join the rows that match, use nulls for those that don't

Summary

- Cartesian join
- . notation
- Inner
- Left Outer join
- Right Outer join
- Full Outer
- Sub query
- Old vs new syntax

Aggregations

Aggregating Functions

```
select * from refdata.instrument;
select count(*),count(ric),count(isin),count(distinct isin)
from refdata.instrument;
```

count excludes nulls. count(*) counts the number of rows

```
select sum(orderqty*price)dollarvalue,count(*)
numCompletedOrders from posttrade.orders where
orderqty=cumqty;
```

show the total value and the number of completed orders

```
select
sum(cumqty),count(*),count(cumqty),floor(avg(cumqty)),floor
(sum(cumqty)/count(cumqty)),floor(sum(cumqty)/count(*))
from posttrade.orders;
```

avg excludes nulls

```
select max(cumqty*price) from posttrade.orders;
```

largest order

Binning

```
select count(*)numInstruments,mic from refdata.instrument
group by mic;
```

How many instruments does each exchange have? Include mic, automatically uses the bucket name, and not the raw column which would be too long

```
select
count(*)numCompletedOrders,sum(orderqty*price)dollarvalue
from posttrade.orders where orderqty=cumqty group by side;
```

1 row per side showing the number of complete orders and their value

```
select
side,count(*)numCompletedOrders,sum(orderqty*price)dollarvalu
e from posttrade.orders where orderqty=cumqty group by side;
```

more helpful if we include the side column

```
select
side,instrument,count(*)numfilledorders,sum(orderqty*price)do
llarvalue
from posttrade.orders where orderqty=cumqty
group by side,instrument;
```

Bucket per side per instrument

```
select rootordid,ordstatus,count(*)cnt from posttrade.orders
group by rootordid,ordstatus having count(*)>5;
```

show the number of rows for each order staturs within each order, exclude any groups with fewer than 6 rows

Set Operations

```
select sum(numrows) from (
 select count(*)numrows from refdata.mic
 union all
 select count(*)numrows from refdata.instrument
 union all
 select count(*)numrows from refdata.fix
 union all
 select 20 numrows from dual);
```

union appends tables that have the same columns as each other

```
select sum(numrows) from (
 select count(*)numrows from refdata.mic
 union
 select count(*)numrows from refdata.instrument
 union
 select count(*)numrows from refdata.fix
 union
 select 20 numrows from dual);
```

```
select * from refdata.instrument where isin in(select isin from refdata.instrument where mic='XLON' intersect select isin from refdata.instrument where
mic='XETR');
```

Find all the products for companies that are listed on both London and German exchanges

```
select * from refdata.instrument where isin in(select isin from refdata.instrument where mic='XLON' minus select isin from refdata.instrument where
mic='XETR');
```

Show all the london instruments that are not also listed on Xetra

Exercise

1. Do a Cartesian join between MIC and INSTRUMENT
2. Filter the above query to make it an equi join
3. Change it again to be a right join, with MIC being on the right(enrich MIC with INSTRUMENT)
4. What is the average number of fills required to complete an order?
5. OPTIONAL: Explain the numbers from the having example?

Changing Data

Create/Alter/Commit/Update

```
CREATE TABLE tablename ( colname1 coltype1, ...);
```

syntax to create a new table

```
create table dhodgins_md_trade(instrument varchar2(6
byte),time timestamp,tradesize number(*,0),price float(126));
```

--TODO what does 126 mean?

Create a new table with 4 columns

```
insert into
dhodgins_md_trade(instrument,time,tradesize,price) values
(2,to_timestamp('2016.08.22 08:00:00.00','YYYY.MM.DD
HH24:MI:SS.FF9'),1000,835.9);
```

Insert a row using the long syntax

```
insert into dhodgins_md_trade values
(2,to_timestamp('2016.08.22 08:00:00.0012345','YYYY.MM.DD
HH24:MI:SS.FF9'),1100,835.85);
```

Insert another row but put the values in the correct order so don't need column names

```
select * from dhodgins_md_trade;
```

If you try this you will see an empty table

```
commit;
```

Save the changes so you can see them

```
alter table dhodgins_md_trade add id number(*,0);
```

Add a new numeric column named id

```
select * from dhodgins_md_trade;
update dhodgins_md_trade set id=1 where price=835.9;
```

Give id 1 to the first row

```
update dhodgins_md_trade set id=2 where price=835.85;
```

Give id 2 to the second row

```
select * from dhodgins_md_trade;
```

Delete/Alter/Update

delete from dhodgins_md_trade where id=2;	**Remove any rows with id of 2**
select * from dhodgins_md_trade;	
alter table dhodgins_md_trade add(tradetype varchar2(2 byte),exchange_time timestamp);	**Add 2 columns to our table**
update dhodgins_md_trade set exchange_time=time-interval '0.1' second;	**Pretend the latency is 100ms**
select * from dhodgins_md_trade;	
alter table dhodgins_md_trade drop column tradetype;	**Delete the tradetype column from our table**
~~alter table dhodgins_md_trade set unused column exchange_time;~~	**~~Hide the exchange_time column~~**
~~select * from dhodgins_md_trade;~~	
~~alter table dhodgins_md_trade drop unused columns;~~	**~~Delete any hidden columns~~**
~~select * from dhodgins_md_trade;~~	

Column Rules(constraints)

```
create table dhodgins_md_trade2(id number(*,0) generated always as identity,
  instrument number(*,0),
  time timestamp,
  seqnum number(*,0),
  side number(*,0),

  constraint dhodgins_md_trade2_instr_fk foreign key(instrument)references refdata.instrument(id),
  constraint dhodgins_md_trade2_timestamp check(time is not null),
  constraint dhodgins_md_trade2_seqnum unique(seqnum),
  constraint dhodgins_md_trade2_side check(side in(1,2,5)));
```

Here constraints are on their own line constraint and have helpful names

```
--Trainer switches to refdata user and does:
grant references on refdata.instrument to delegate;
--Switch back to delegate window and retry
insert into dhodgins_md_trade2(instrument,time,seqnum,side)values(-1,to_timestamp('2016.08.22
08:00:00.00','YYYY.MM.DD HH24:MI:SS.FF9'),0,1);

insert into dhodgins_md_trade2(instrument,time,seqnum,side)values(1,to_timestamp('2016.08.22
08:00:00.00','YYYY.MM.DD HH24:MI:SS.FF9'),0,1);

select * from dhodgins_md_trade2;
```

Omit autogenerated column, nicer error that when constraint had no name

Automated Queries(Linux)

```
cat ~dhodgins/teach/queryOracle.sh
```

```
export PATH=$PATH:~dhodgins/teach
```

```
queryOracle.sh delegate/pass@10.20.40.53/oradb1 "select * from refdata.instrument"
```

Exercise

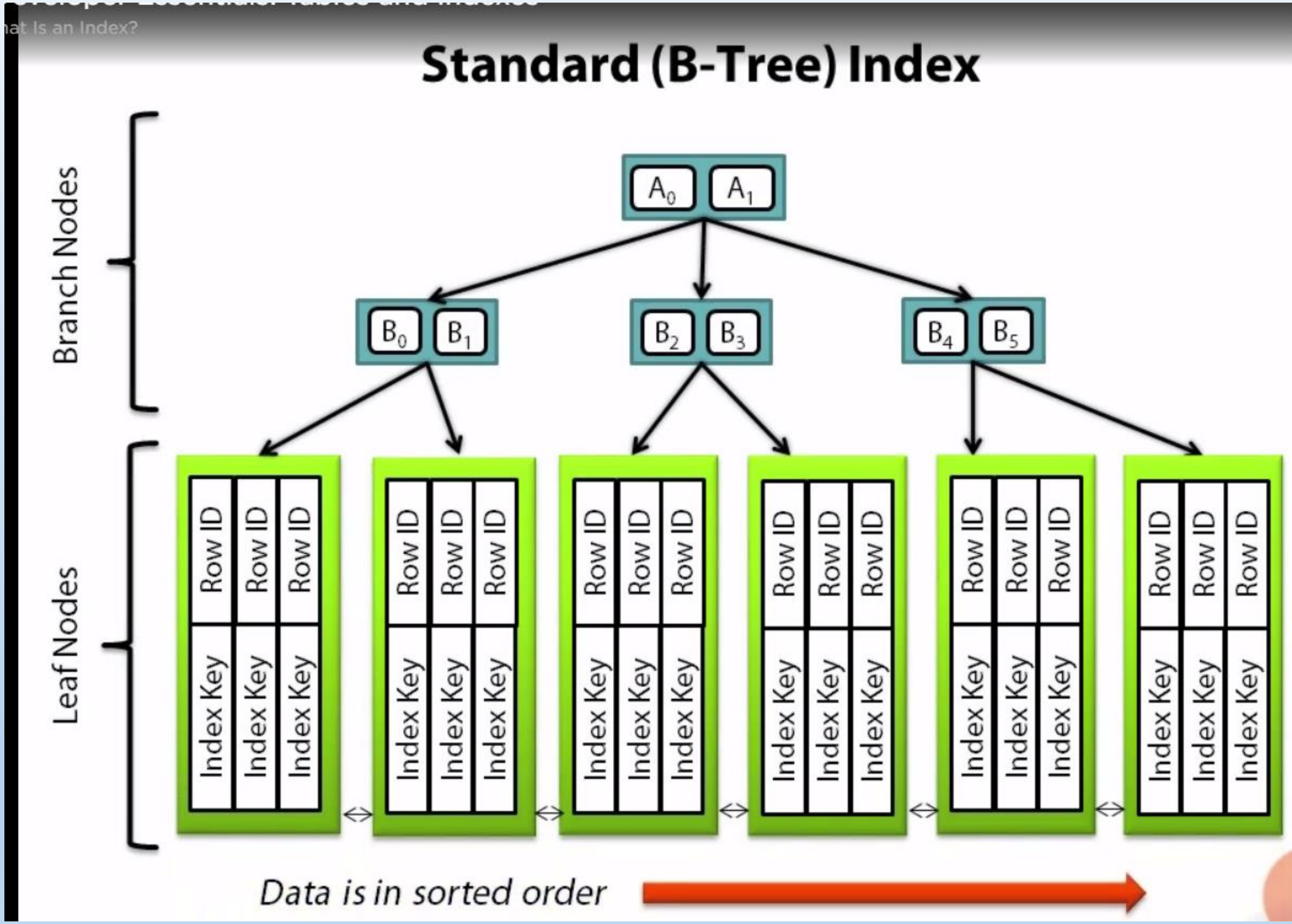
1. Create a new table named userX_ftse100 with columns 'name','price','volume', add appropriate constraints for each column
2. populate this with all the companies starting with 'G'
3. Change the table and add a column called 'market_cap'
4. Populate the 'market_cap' column
5. Combine userX+1 and userX-1 tables into a single new table with an additional column called 'src_table' which indicates which table each row came from

Advanced Oracle

- Indexes
- Views
- Query Optimisation
- PL/SQL

Indexes

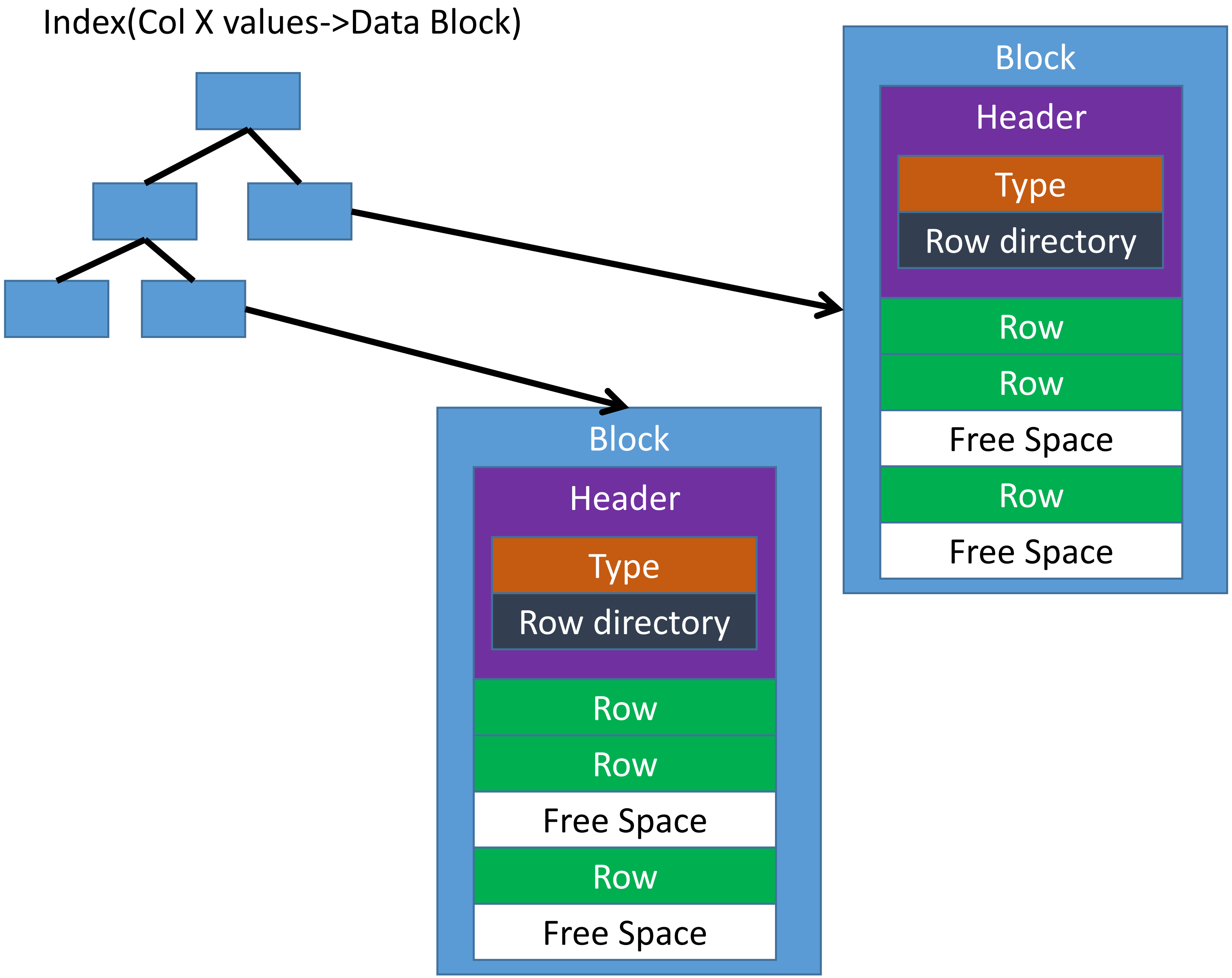
- what are A0, A1?
- Values < A0 are in left child, between A0 and A1 in middle, and >A1 in right
- similarly for B, but the diagram only has 2 leaf nodes for some reason



hat Is an Index?

Standard (B-Tree) Index

Indexes + Data Blocks



1. Traverse index(log n), n=# unique values in column
2. Read block header row directory(1)
3. Read rows(m), m=number of rows to read

Best case you have a tiny number of unique values in your index column

O(1)+O(m), where m is number of rows being queried

Worst case you wouldn't create an index on a column that had more than log(t) unique values, where t is the total number of rows

O(log(log(t))+m)

Views

- Why
  - hide complexity
  - allow change per team
  - performance(caching result)
- SQL ....
- unlike regular selects views complain about duplicate column names
- using select * only captures the column names once and does not reflect any changes to the schemas going forwards.
- show them a view and then show a proper view using stored procs

- -- validation select distinct startdate,enddate from
- --validate select min(startdate),max(enddate) from

```
-- where md.trade_deriv.sym='EDU16'

--todo replace * with col names

--don't run we already did this --

create or replace view md.trade_deriv_with_ref_view as

  select DT,TIME,EXCHTIME,md.trade_deriv.SYM,PRICE,md.trade_deriv.SZE,refdata.product.productCode,STARTDATE,ENDDATE,NAME,EXCHAnge,CATEGORY,SUBCATEGORY,CLASS,refdata.product.SZE delivery_size
from

    md.trade_deriv left outer join refdata.contract

      on md.trade_deriv.sym=refdata.contract.sym

    left outer join refdata.product

      on refdata.product.productcode=refdata.contract.productcode;


select * from md.trade_deriv;

select * from md.trade_deriv_with_ref_view;
```

Views

- with grant option
  - needed if you want to grant a view on someone elses table
- select * from all_views where owner='MD';