
How to do Almost Everything
(to Administer Db2)
via SQL and Jupyter Notebook
Ember Crooks
Session code:
IDUGDb2
Db2

Agenda

- Work with Db2 privileges and authorities via SQL
- Delve into the details of tables with SQL
- Use the MON_GET* table functions and views to understand what is going on in a Db2 database
- Use SQL to dig into the details of how your system and database is configured

Why SQL?

- Use any language, any client
- Containers = avoid SSH
- Use the same interface for containerized and non-containerized environments
- Vast majority of DBA tasks possible

Why Jupyter Notebook?

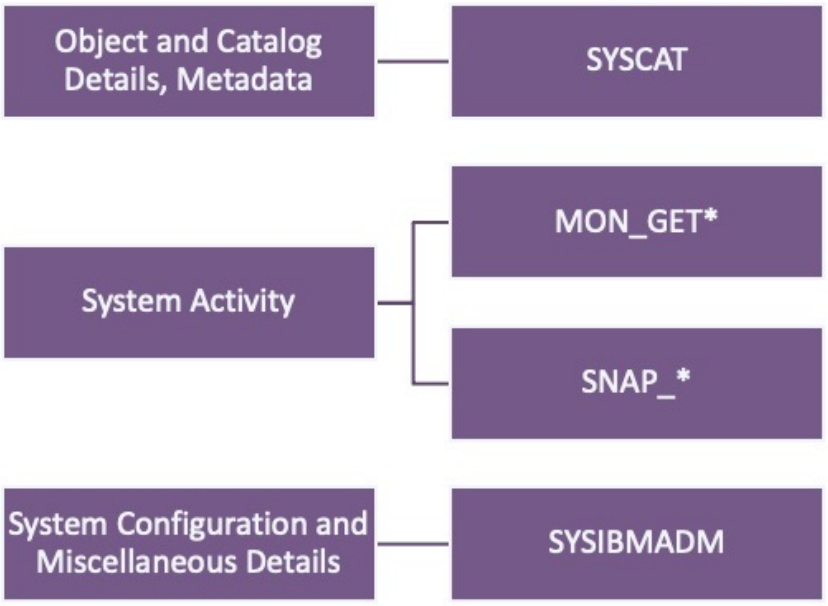
- Popular in data science
- Nice and easy visualizations
- Easy SQL with the SQL Magic
- Incorporate other elements
- Combine explanatory text with executable SQL
- Easy to turn exploratory notebook into python script in whole or part
- Skills transferable

3

Jupyter Notbook

- Learn more with Introduction to Using Jupyter Notebook with Db2
- Download the notebook for this presentation from my GitHub repo

Using SQL to Investigate Db2



Work with Db2 Privileges and Authorities via SQL

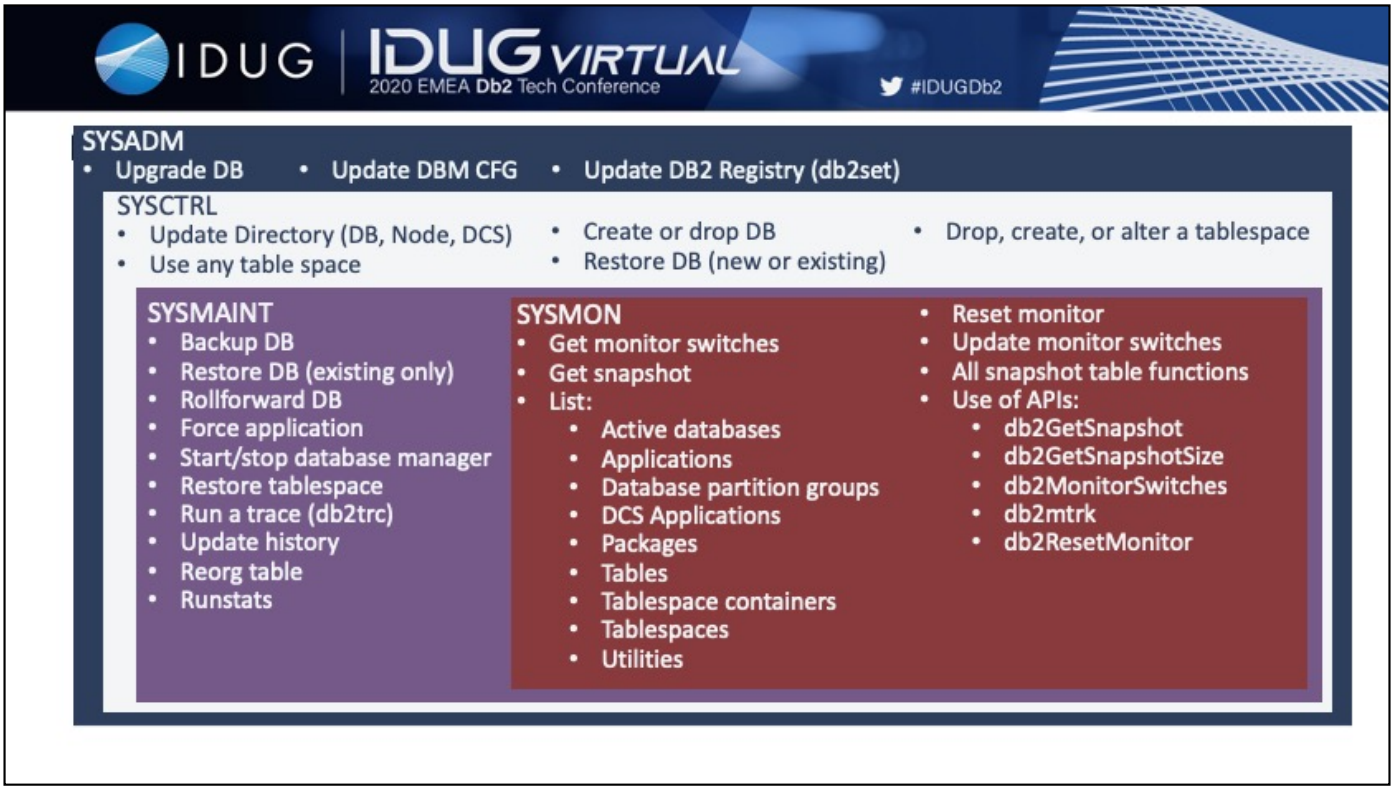
How Permissions are Conferred

User	Role	Group	PUBLIC
• Exists at OS or authentication level • User does not have to exist • Permissions can be granted directly or inherited from a role, group, or PUBLIC	• Exists purely within DB2 • Role explicitly created within DB2 • Permissions are granted to the role • The role is granted to a user or a group	• Exists at OS or authentication level • Group does not have to exist • Permissions can be granted to a group	• All users that can authenticate have these permissions • Can be dangerous • Revoke connect privilege on the database from PUBLIC

User and group naming rules:
https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.1.0/com.ibm.db2.luw.admin.dbobj.doc/doc/c0007248.html

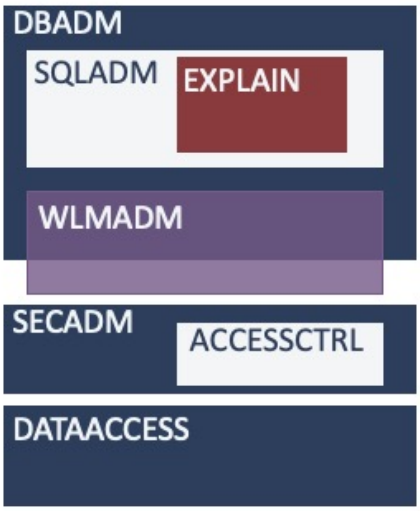
Group vs. Role

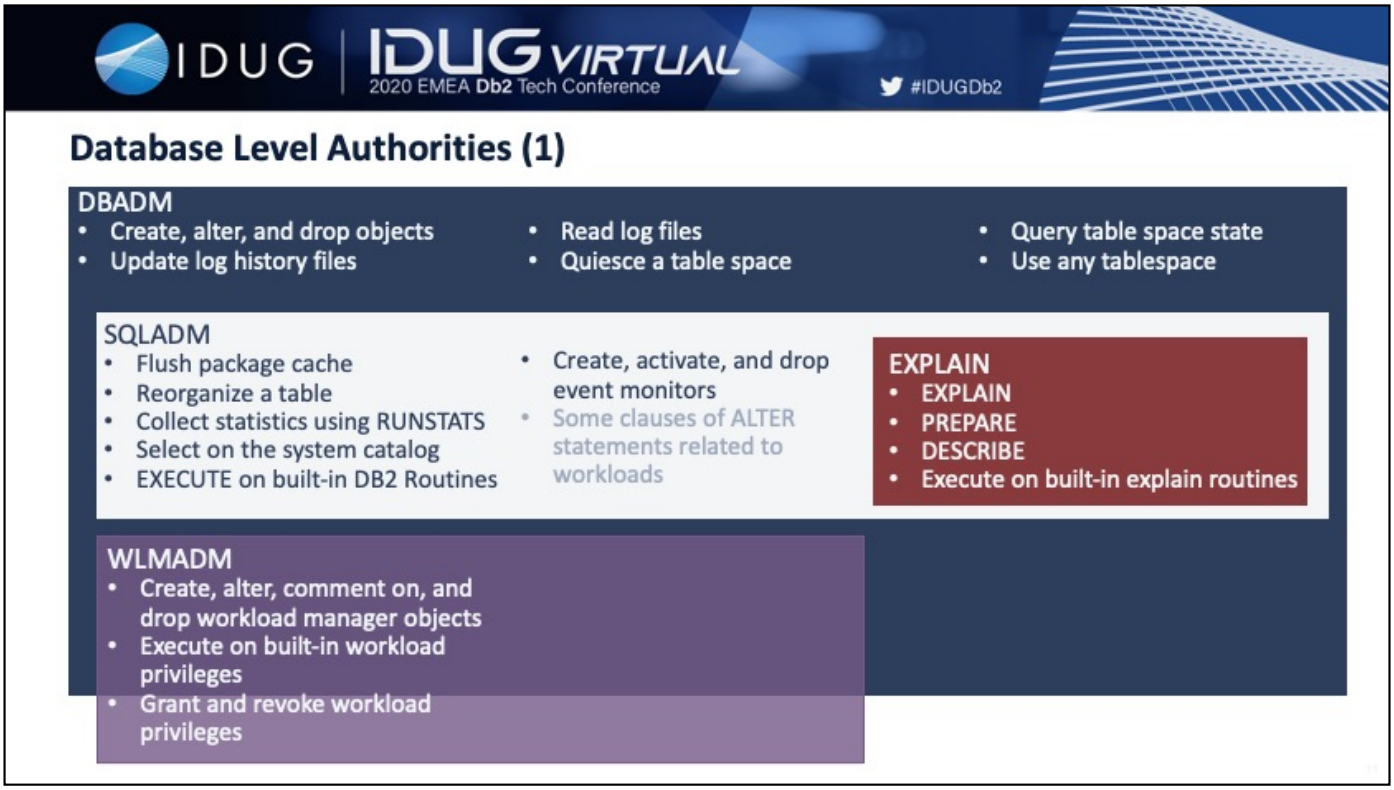
Group	Role
• Exist at the OS or security authority level • Any permission that can be granted to a user can also be granted to a group • Each user that is a member of the group (at the OS level) holds the Permissions granted to the role	• Exist within DB2 • Any permission that can be granted to a user can also be granted to a group • Each user assigned a role holds the Permissions granted to the role • Controlled by the DBA • Does not rely on OS/LDAP resolution



Some of these actions can be accomplished using database-level authorities or object-level permissions, too.

Overview of Database Level Authorities





Some of these actions can be accomplished using database-level authorities or object-level permissions, too.

Database Level Authorities (2)

SECADM
- Create, alter, comment on, and drop security objects
- GRANT and REVOKE database privileges and authorities
- EXECUTE built-in audit routines
- GRANT EXECUTE on built-in audit routines
- Change ENCRLIB and ENCROPTS db cfg parameters
- EXECUTE built-in encryption and key rotation routines
- AUDIT a database or database object
- Use TRANSFER OWNERSHIP when not the owner of the object

ACCESSCTRL
- Grant and revoke EXPLAIN, SQLADM, and WLMADM
- Grant and revoke BINDADD, CONNECT, CREATETAB, CREATE_EXTERNAL_ROUTINE, CREATE_NOT_FENCED_ROUTINE, IMPLICIT_SCHEMA, LOAD, and QUIESCE_CONNECT
- Grant and revoke all privileges on tables, indexes, global variables, nicknames, packages, routines (except audit), schemas, sequences, servers, table spaces, views, and XSR objects
- Select on the system catalog

DATAACCESS
- LOAD
- SELECT, UPDATE, INSERT, and DELETE on all tables, views, MQTs, and nicknames
- READ on all global variables
- EXECUTE on all packages, modules and retunes not related to security
- USAGE on all sequences and XSR objects

Investigating Database-Level Authorities and Permissions

```
select grantee
from syscat.dbauth
where connectauth='Y'
```

```
GRANTEE
--------------------
PUBLIC

  1 record(s) selected.
```

```
select dbadmauth
         , connectauth
         , dataaccessauth
from syscat.dbauth
where grantee='ECROOKS'
```

```
DBADMAUTH CONNECTAUTH DATAACCESSAUTH
--------- ----------- --------------
Y         N           Y

  1 record(s) selected.
```

Querying Object Permissions (1|2)

```
select substr(tabschema,1,8) as tabschema
        , substr(tabname,1,18) as tabname
        , controlauth
        , deleteauth
        , insertauth
        , selectauth
        , updateauth
from syscat.tabauth
where grantee='ECROOKS'
```

```
TABSCHEMA TABNAME            CONTROLAUTH DELETEAUTH INSERTAUTH SELECTAUTH UPDATEAUTH
--------- ------------------ ----------- ---------- ---------- ---------- ----------
DB2INST1  EMPLOYEE           Y           G          G          G          G
DB2INST1  SALES              N           N          N          Y          N

  2 record(s) selected.
```

Querying Object Permissions (2|2)

```
select substr(grantee,1,8) as grantee
        , controlauth
        , deleteauth
        , insertauth
        , selectauth
        , updateauth
from syscat.tabauth
where tabschema='DB2INST1'
        and tabname='SALES'
```

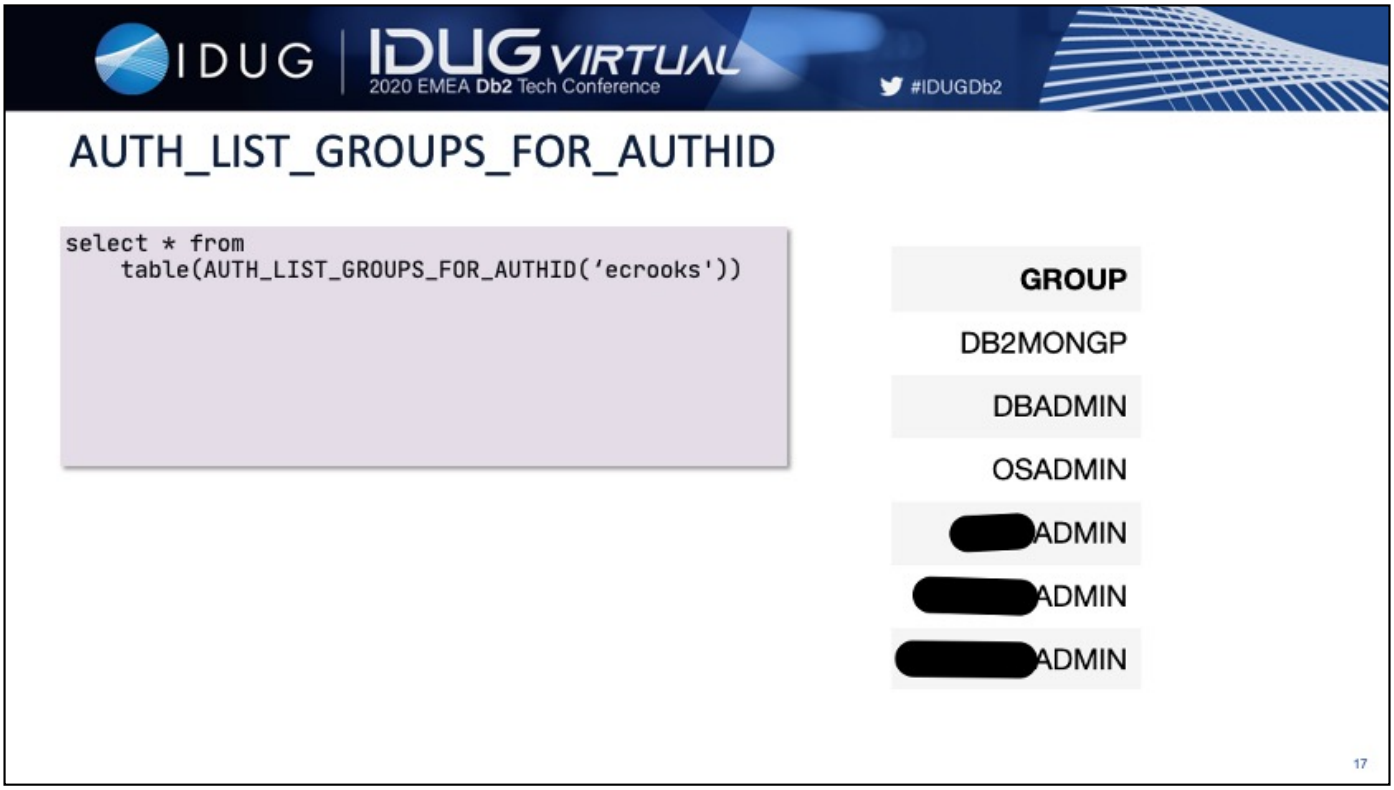
```
GRANTEE  CONTROLAUTH DELETEAUTH INSERTAUTH SELECTAUTH UPDATEAUTH
-------- ----------- ---------- ---------- ---------- ----------
DB2INST1 Y           G          G          G          G
ECROOKS  N           N          N          Y          N

  2 record(s) selected.
```

Querying All Object Permissions

```
select *
from sysibmadm.privileges
where authid='DB2ADMIN'
```

authid	authidtype	privilege	grantable	objectname	objectschema	objecttype
DB2ADMIN	U	EXECUTE	Y	CADMSETP	NULLID	DB2 PACKAGE
DB2ADMIN	U	BIND	Y	CADMSETP	NULLID	DB2 PACKAGE
DB2ADMIN	U	CONTROL	N	CADMSETP	NULLID	DB2 PACKAGE
DB2ADMIN	U	EXECUTE	Y	EXPLAIN_GET_MSGS	DB2ADMIN	FUNCTION
DB2ADMIN	U	USE	Y	SYSTOOLSPACE		TABLESPACE
DB2ADMIN	U	CONTROL	N	ARG_I1	DB2ADMIN	INDEX
DB2ADMIN	U	CONTROL	N	EXP_DIAG_DAT_I1	DB2ADMIN	INDEX
DB2ADMIN	U	CONTROL	N	IDX_I1	DB2ADMIN	INDEX



https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.1.0/com.ibm.db2.luw.sql.rtn.doc/doc/r0021976.html

For Roles: AUTH_LIST_GROUPS_FOR_AUTHID

AUTH_LIST_AUTHORITIES_FOR_AUTHID

```
select * from
    table(AUTH_LIST_AUTHORITIES_FOR_AUTHID('ecrooks','U'))
```

authority	d_user	d_group	d_public	role_user	role_group	role_public	d_role
SYSADM	*	N	*	*	*	*	*
DBADM	N	Y	N	N	N	N	*
CREATETAB	N	N	Y	N	N	N	*
BINDADD	N	N	Y	N	N	N	*
CONNECT	N	Y	Y	N	N	N	*
CREATE_NOT_FENCED_ROUTINE	N	N	N	N	N	N	*
SYSCTRL	*	N	*	*	*	*	*
SYSMAINT	*	N	*	*	*	*	*
IMPLICIT_SCHEMA	N	N	Y	N	N	N	*

Complicated SQL for Nicely Formatted Output (1|2)

```
with tab_perms as(
select rtrim(grantee) as grantee
    , granteetype
    , count(*) as tab_write_access
from syscat.tabauth ta
where ta.updateauth='Y'
    or ta.insertauth='Y'
    or ta.deleteauth='Y'
group by grantee, granteetype
) ,
tab_select as(
select rtrim(grantee) as grantee
    , granteetype
    , count(*) as tab_read_access
from syscat.tabauth ta
where ta.selectauth='Y'
group by grantee, granteetype
)
select coalesce(da.grantee, tp.grantee) as grantee
    , coalesce(da.granteetype, tp.granteetype) as granteetype
    , securityadmauth
    , dbadmauth
    , dataaccessauth
    , case dataaccessauth when 'Y' then (select count(*) from syscat.tables where type in ('T','V')) else tp.tab_write_access end as
tab_write_access
    , case dataaccessauth when 'Y' then (select count(*) from syscat.tables where type in ('T','V')) else ts.tab_read_access end as
tab_read_access
from syscat.dbauth da
    left outer join tab_perms tp on da.grantee=tp.grantee and da.granteetype=tp.granteetype
    left outer join tab_select ts on da.grantee=ts.grantee and da.granteetype=ts.granteetype
where securityadmauth='Y' or dbadmauth='Y' or dataaccessauth='Y' or tab_write_access > 0 or tab_read_access > 0
order by granteetype desc, securityadmauth desc, dbadmauth desc, dataaccessauth desc, tab_write_access desc, tab_read_access desc
with ur
```

19

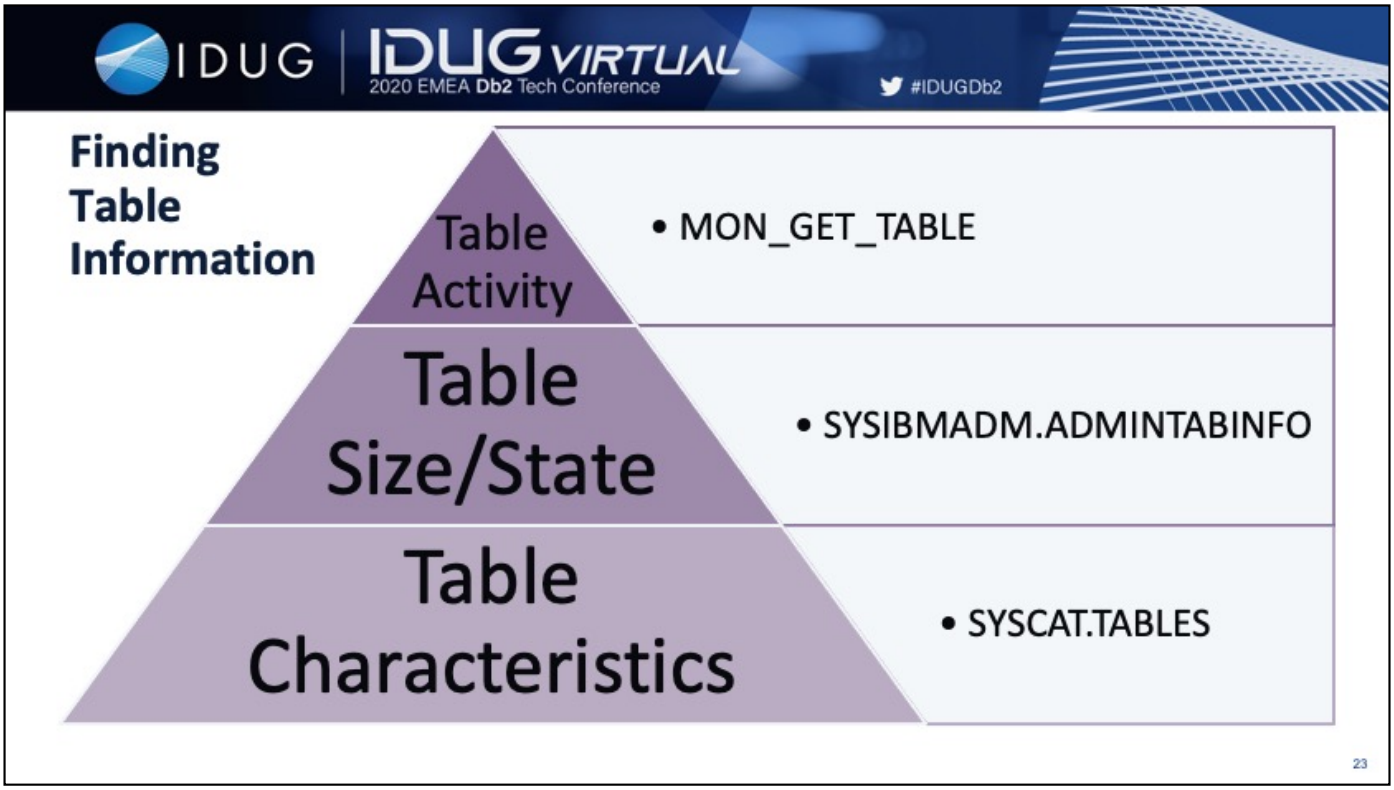
Complicated SQL for Nicely Formatted Output (2|2)

grantee	granteetype	securityadmauth	dbadmauth	dataaccessauth	tab_write_access	tab_read_access
DB2INST1	U	Y	Y	Y	1752	1752
[redacted]USR	U	N	N	Y	1752	1752
DOMO	U	N	N	N	None	11
[redacted]DBADMIN	G	N	Y	Y	1752	1752
[redacted]DBRO	G	N	N	N	None	1173
[redacted]DBRW	G	N	N	N	1098	1173
PUBLIC	G	N	N	N	10	416
DB2MONGP	G	N	N	N	2	9

Complicated SQL for Nicely Formatted Output – Add LDAP!

	granteetype	securityadmauth	dbadmauth	dataaccessauth	tab_write_access	tab_read_access	group_members
db2inst1	U	Y	Y	Y	4232.0	4232	NaN
db2mongp	G	N	N	N	2.0	4	[redacted]Ember Crooks]
[redacted]dbadmin	G	N	Y	Y	4232.0	4232	[redacted]Ember Crooks]
domo	U	N	N	N	NaN	14	NaN
oms	U	N	N	N	17.0	17	NaN
test439	U	N	N	N	NaN	1	NaN
[redacted]dbro	G	N	N	N	NaN	3232	[redacted]
[redacted]dbrw	G	N	N	N	2436.0	3421	[redacted]
[redacted]usr	U	N	N	Y	4232.0	4232	NaN

Delve into the details of tables with SQL



SYSIBMADM.ADMINTABINFO takes time to query, particularly for all tables, as it is actually calculating information rather than just returning information that is stored on disk

Unused Tables

```
select  t.lastused
        , date(stats_time) as stats_time
        , date(create_time) as create_time
        , substr(t.tabschema,1,10) as tabschema
        , substr(t.tabname,1,25) as tabname
        , bigint(card) as table_card
        , mt.table_scans
        , mt.rows_read
        , mt.rows_inserted + mt.rows_updated + mt.rows_deleted as rows_altered
        , t.volatile
        , mt.member
    from    syscat.tables t
        join table(mon_get_table('','',-2)) as mt on t.tabschema=mt.tabschema and t.tabname = mt.tabname
    where
        t.tabschema not like 'SYS%'
        and t.tabname not like '%EXPLAIN%'
        and t.tabname not like '%ADVISE%'
        and t.lastused < current date - 30 days
        and type = 'T'
    order by t.lastused, t.card desc, t.tabschema, t.tabname
    with ur
```

lastused	stats_time	create_time	tabschema	5	6	tabname	8	9	table_card	table_scans	rows_read	rows_altered	VOLATILE	MEMBER
2016-11-06	2020-09-04	2016-11-06	WSCOMUSR	1	10	X_TMPIITEM	1	25	34	0	340	0		0
2016-11-06	2020-09-04	2016-10-06	WSCOMUSR	1	10	X_TMPEXPCNT	1	25	12	0	120	0		0
2016-11-06	2020-09-04	2016-11-06	WSCOMUSR	1	10	X_TMPIITEMLIST	1	25	11	0	110	0		0
2018-10-15	2020-09-04	2016-10-06	WSCOMUSR	1	10	X_TMPCPRCNT	1	25	7	0	70	0		0
2019-04-10	2020-09-04	2019-04-10	BDC837	1	10	ACCOUNTS	1	25	1889106	0	18891060	0		0
2019-08-05	2020-09-04	2019-08-02	WSCOMUSR	1	10	STLOCIDS	1	25	9066	0	90660	0		0

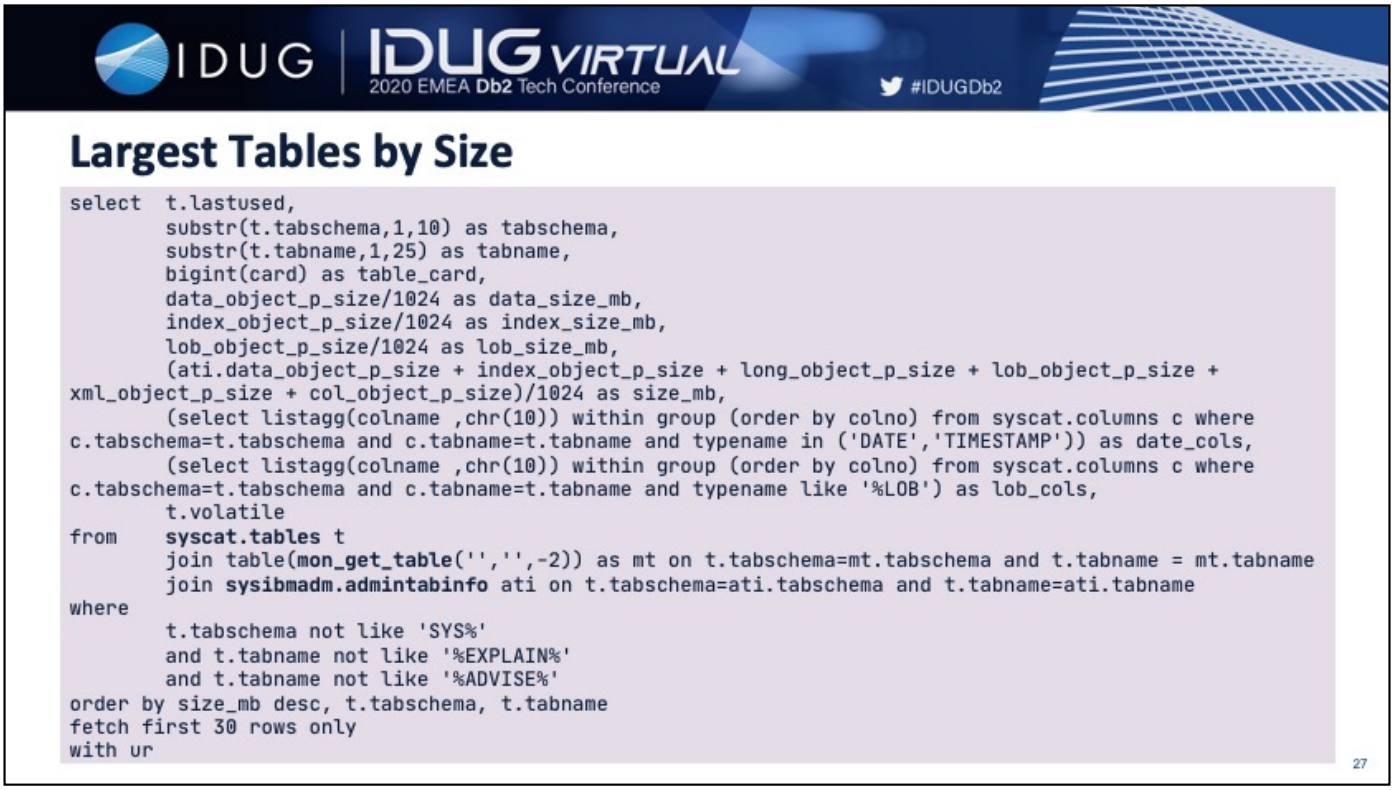
Most Used tables

```
select  t.lastused
        , substr(t.tabschema,1,10) as tabschema
        , substr(t.tabname,1,25) as tabname
        , bigint(card) as table_card
        , mt.table_scans
        , mt.rows_read
        , case when card >0 then mt.rows_read/card else 0 end as avg_reads_per_row
        , mt.rows_inserted + mt.rows_updated + mt.rows_deleted as rows_altered
        , t.volatile
        , mt.member
    from    syscat.tables t
        join table(mon_get_table('','',-2)) as mt on t.tabschema=mt.tabschema and
t.tabname = mt.tabname
    where
        t.tabschema not like 'SYS%'
        and t.tabname not like '%EXPLAIN%'
        and t.tabname not like '%ADVISE%'
    order by 7 desc, 8 desc, t.tabschema, t.tabname, mt.member
    fetch first 20 rows only
    with ur
```

Most Used Tables – Applying Additional Python Formatting

```
busy_tab_df=mostused_tables.DataFrame()
busy_tab_df['table_card'] = busy_tab_df.apply(lambda x: "{:,}".format(x['table_card']), axis=1)
busy_tab_df['table_scans'] = busy_tab_df.apply(lambda x: "{:,}".format(x['table_scans']), axis=1)
busy_tab_df['rows_read'] = busy_tab_df.apply(lambda x: "{:,}".format(x['rows_read']), axis=1)
busy_tab_df['avg_reads_per_row'] = busy_tab_df.apply(lambda x: "{:,}".format(x['avg_reads_per_row']), axis=1)
busy_tab_df['rows_altered'] = busy_tab_df.apply(lambda x: "{:,}".format(x['rows_altered']), axis=1)
display(HTML(busy_tab_df.to_html(index=False)))
```

lastused	tabschema	tabname	table_card	table_scans	rows_read	avg_reads_per_row	rows_altered	VOLATILE	MEMBER
2020-09-04	WSCOMUSR	SCHACTIVE	22	24	6,108,551	277,661	57,455		0
2020-09-03	WSCOMUSR	ACACTION	3,256	96,514	314,332,127	96,539	2,822		0
2020-09-04	WSCOMUSR	XSTORECONF	673	16	9,964,814	14,806	69		0
2020-09-04	WSCOMUSR	LANGUAGE	14	8,319	119,475	8,533	14		0
2020-09-04	WSCOMUSR	TRANSPORT	22	5,927	135,260	6,148	17		0
2020-09-04	WSCOMUSR	BUSCHN	6	6,054	36,396	6,066	0		0
2020-09-04	WSCOMUSR	BUSCHNDESC	6	6,011	36,138	6,023	0		0
2020-09-04	WSCOMUSR	STORE	7	852	41,385	5,912	2		0



Because this query is accessing ADMINTABINFO for all tables, it takes a while to run.

Largest Tables by Size – Partial Sample output

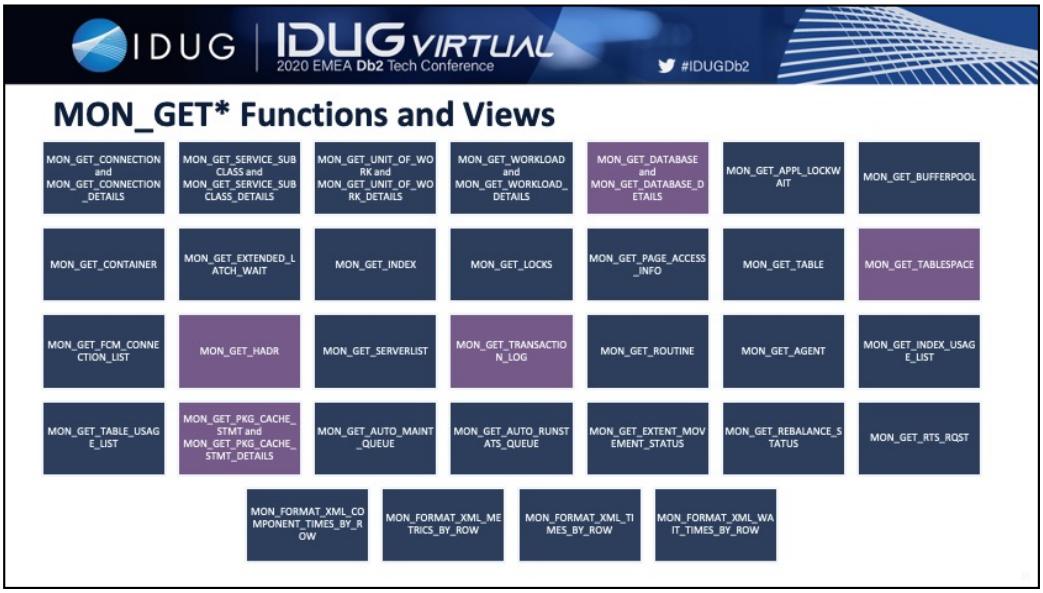
lastused	tabschema	tabname	table_card	data_size_mb	index_size_mb	lob_size_mb	size_mb	date_cols	lob_cols	VOLATILE
2020-08-25	[redacted]	MSGARCHIVE	1909681	226	89	108564	108880	MSGLASTUPDATE MSGDATECREATED	MESSAGE	
2020-09-07	[redacted]	BUSEVENT	7050315	20885	235	0	21120	CREATETSTMP	None	
2020-09-07	[redacted]	XIITEM	16060124	2205	822	6746	9774	LASTUPDATE	IITEMVALUE	
2020-09-07	[redacted]	INVAVL	38894460	4471	2725	0	7196	AVAILTIME LASTUPDATE	None	
2020-09-07	[redacted]	USERREG	4554546	1507	1505	0	3012	PASSWORDCREATION PASSWORDINVALID	None	
2020-09-06	[redacted]	USERS	4554558	1207	1600	0	2808	LASTORDER REGISTRATION LASTSESSION REGISTRATIONUPDATE REGISTRATIONCANCEL PREVLASTSESSION	None	

Use the MON_GET* table functions and views to understand what is going on in a Db2 database

Why MON_GET*

- Snapshots are deprecated
- Lightweight in-memory monitoring
- Thousands of metrics

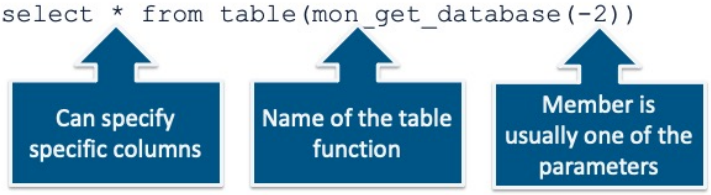
- No reset functionality!



SQL interface to lightweight in-memory monitoring

Querying MON_GET* Table Functions

```
select * from table(mon_get_database(-2))
```



- Variable number of parameters depending on the object
- For example, mon_get_table requires tabschema and tabname
- Can use '' or NULL for many of the parameters to return data on all objects

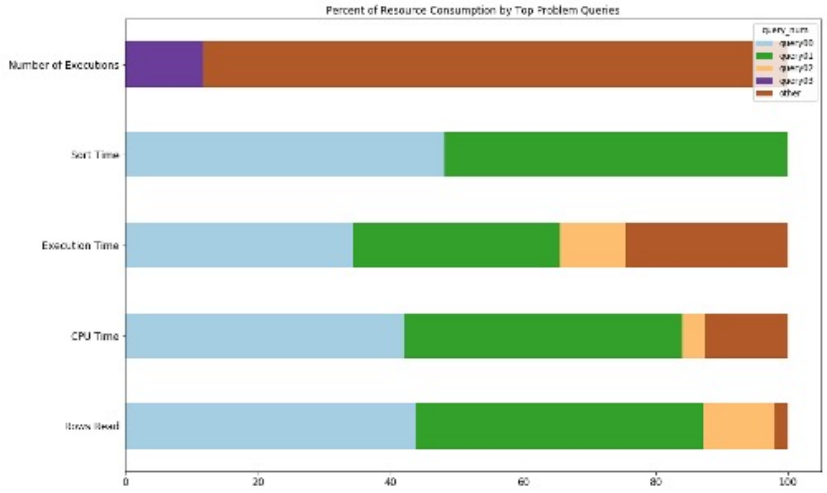
Finding Problem SQL (DB2 9.7 and up)

```
WITH SUM_TAB (SUM_RR, SUM_CPU, SUM_EXEC, SUM_SORT, SUM_NUM_EXEC) AS (
        SELECT  FLOAT(SUM(ROWS_READ)),
                FLOAT(SUM(TOTAL_CPU_TIME)),
                FLOAT(SUM(STMT_EXEC_TIME)),
                FLOAT(SUM(TOTAL_SECTION_SORT_TIME)),
                FLOAT(SUM(NUM_EXECUTIONS))
            FROM TABLE(MON_GET_PKG_CACHE_STMT ( 'D', NULL, NULL, -2)) AS T
        )
SELECT  SUBSTR(STMT_TEXT,1,10) as STATEMENT,
        ROWS_READ,
        DECIMAL(100*(FLOAT(ROWS_READ)/SUM_TAB.SUM_RR),5,2) AS PCT_TOT_RR,
        TOTAL_CPU_TIME,
        DECIMAL(100*(FLOAT(TOTAL_CPU_TIME)/SUM_TAB.SUM_CPU),5,2) AS PCT_TOT_CPU,
        STMT_EXEC_TIME,
        DECIMAL(100*(FLOAT(STMT_EXEC_TIME)/SUM_TAB.SUM_EXEC),5,2) AS PCT_TOT_EXEC,
        TOTAL_SECTION_SORT_TIME,
        DECIMAL(100*(FLOAT(TOTAL_SECTION_SORT_TIME)/SUM_TAB.SUM_SORT),5,2) AS PCT_TOT_SRT,
        NUM_EXECUTIONS,
        DECIMAL(100*(FLOAT(NUM_EXECUTIONS)/SUM_TAB.SUM_NUM_EXEC),5,2) AS PCT_TOT_EXEC,
        DECIMAL(FLOAT(STMT_EXEC_TIME)/FLOAT(NUM_EXECUTIONS),10,2) AS AVG_EXEC_TIME
    FROM TABLE(MON_GET_PKG_CACHE_STMT ( 'D', NULL, NULL, -2)) AS T, SUM_TAB
    WHERE DECIMAL(100*(FLOAT(ROWS_READ)/SUM_TAB.SUM_RR),5,2) > 10
        OR DECIMAL(100*(FLOAT(TOTAL_CPU_TIME)/SUM_TAB.SUM_CPU),5,2) >10
        OR DECIMAL(100*(FLOAT(STMT_EXEC_TIME)/SUM_TAB.SUM_EXEC),5,2) >10
        OR DECIMAL(100*(FLOAT(TOTAL_SECTION_SORT_TIME)/SUM_TAB.SUM_SORT),5,2) >10
        OR DECIMAL(100*(FLOAT(NUM_EXECUTIONS)/SUM_TAB.SUM_NUM_EXEC),5,2) >10
    ORDER BY ROWS_READ DESC FETCH FIRST 20 ROWS ONLY WITH UR
```

Problem SQL Text Output

	STATEMENT	rows_read	pct_tot_rr	total_cpu_time	pct_tot_cpu	stmt_exec_time	pct_tot_exec_time	total_section_sort_time	pct_tot_srt	num_executions
0	SELECT CATENTRY_ID, a.PAR	109,525,340	43.72	120,779,059	42.10	139,563	34.38	74,386	48.16	2
1	SELECT CATENTRY_ID, a.PAR	108,858,532	43.45	120,391,704	41.96	126,543	31.17	79,853	51.70	2
2	SELECT A.FIRSTNAME, A.LAS	27,099,108	10.81	9,625,630	3.35	40,511	9.98	0	0.00	2
3	VALUES (CURRENT SCHEMA)	0	0.00	38,332	0.01	38	0.00	0	0.00	3,487

Comparing Queries Visually



Returning Additional Information (1|4)

```
pd.set_option('display.max_colwidth', -1)
for index, row in df.iterrows():
    # skip the "other" row added to balance out numbers for the metrics
    if row['query_num'] == 'other':
        continue
    # Display basic information about the query
    display(Markdown("## Query "+str(index)))
    display(Markdown("### Query Characteristics"))
    display(Markdown("Executed "+str(row['num_executions'])+" times since last placed in the package cache at "+str(row['insert_timestamp'])))
    display(Markdown("Consumed "+str(row['pct_tot_rr'])+" percent of all rows read by all queries in the package cache."))
    display(Markdown("Consumed "+str(row['pct_tot_cpu'])+" percent of all cpu time used by all queries in the package cache."))
    display(Markdown("Consumed "+str(row['pct_tot_exec_time'])+" percent of all execution time used by all queries in the package cache."))
    display(Markdown("Consumed "+str(row['pct_tot_srt'])+" percent of all sort time used by all queries in the package cache."))
    display(Markdown("### Query Text"))
    formatted_sql=sqlparse.format(df['full_statement'][index], reindent=True)
    print(formatted_sql.replace("\\n","<br>"))
    # If a database connection is available, gather additional information about this query
    ## Note: explain may fail if the interval between running the query to find problem sql and this section was too long, and the section has been cleared
from the package cache
    if conn:
        ##When db connection is available
        display(Markdown("### Query Explain Plan"))
        display(row.dtypes)
        exe_id=row['executable_id']
        ex_schema=user.upper()
        ex_requester=''
        ex_time=''
        src_name=''
        src_schema=schema
        src_version=''
        %sql call explain_from_section(x'{exe_id}', 'M', NULL, 0, :ex_schema, :ex_requester, :ex_time, :src_name, :src_schema, :src_version)
        expln_plan=%sql select * from {user}.last_explained
        print(expln_plan)
```

Returning Additional Information (2|4)

Query 0

Query Characteristics

Executed 2 times since last placed in the package cache at 2020-09-04 18:50:28.127996

Consumed 43.72 percent of all rows read by all queries in the package cache.

Consumed 42.1 percent of all cpu time used by all queries in the package cache.

Consumed 34.38 percent of all execution time used by all queries in the package cache.

Consumed 48.16 percent of all sort time used by all queries in the package cache.

Returning Additional Information (3|4)

Query Text

```
SELECT CATENTRY_ID,
       a.PARTNUMBER as PARTNUMBER,
       COLORTYPE,
       COLORNUMBER,
       COLORNAME,
       RGBCOLOR,
       PONUMBER,
       custnum as account_numbers,
       frequency,
       lastpurchased,
       null as PRODUCTLINEID,
       null as PRODUCTLINENAME,
       null as PRODUCTLINETHUMBNAIL,
       CONCAT('Item_', ROW_NUMBER() OVER()) AS ID
from catentry a,
     xfreqpurprods b
where a.partnumber = b.partnumber
union
SELECT C.CATENTRY_ID_parent as catentry_id,
       a.PARTNUMBER AS PARTNUMBER,
       COLORTYPE,
       COLORNUMBER,
       COLORNAME,
       RGBCOLOR,
       PONUMBER,
       custnum as account_numbers,
       frequency,
       lastpurchased,
```

Returning Additional Information (4|4)

Explain Plan

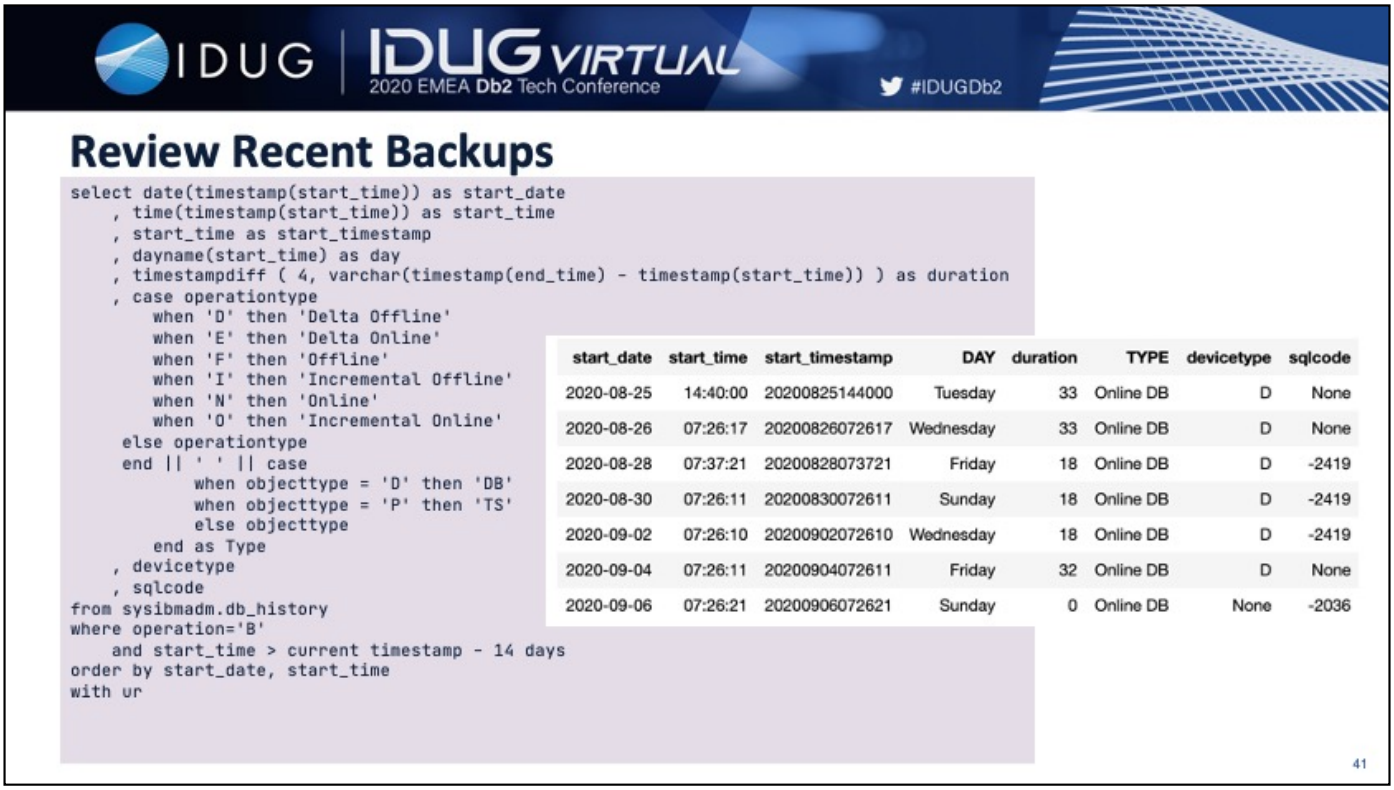
ID	Operation	Rows	Cost
1	RETURN		151049712
2	TBSCAN	18913040 of 18913040 (100.00%)	151049712
3	SORT (UNIQUE)	18913040 of 18913040 (100.00%)	134343520
4	UNION	18913040 of 5558683	6108816
5	HSJOIN	5558683 of 93345	3066507
6	HSJOIN	93345 of 93345 (100.00%)	41710
7	TBSCAN CATENTRY	224255 of 224255 (100.00%)	13612
8	HSJOIN	93345 of 93345 (100.00%)	28086
9	TBSCAN CATENTRY	224255 of 224255 (100.00%)	13612
10	NLJOIN	93345 of 1	14464
11	TBSCAN	93345 of 93345 (100.00%)	1699
12	SORT	93345 of 93345 (100.00%)	1699
13	IXSCAN SQL150812163059940	93345 of -1	1681
14	FETCH CATENTDESC	1 of 1 (100.00%)	13
15	IXSCAN SQL150812163059930	1 of 332144 (.00%)	6
16	TBSCAN XFREQPURPRODS	13354357 of 13354357 (100.00%)	3024001
17	HSJOIN	13354357 of 224255	3038453
18	TBSCAN CATENTRY	224255 of 224255 (100.00%)	13612
19	TBSCAN XFREQPURPRODS	13354357 of 13354357 (100.00%)	3024001

Predicate Information

Predicate		Filter
5	JOIN (Q5.PARTNUMBER = Q4.PARTNUMBER)	00.00
6	JOIN (Q5.CATENTRY_ID = Q3.CATENTRY_ID_CHILD)	00.00
8	JOIN (Q3.CATENTRY_ID_PARENT = Q2.CATENTRY_ID)	00.00
10	JOIN (Q1.CATENTRY_ID = Q3.CATENTRY_ID_PARENT)	00.00
15	START (Q1.CATENTRY_ID = Q3.CATENTRY_ID_PARENT)	00.00
	START (Q1.LANGUAGE_ID = -1)	67.51
	STOP (Q1.CATENTRY_ID = Q3.CATENTRY_ID_PARENT)	00.00
	STOP (Q1.LANGUAGE_ID = -1)	67.51
17	JOIN (Q11.PARTNUMBER = Q10.PARTNUMBER)	00.00

Explain plan (c) 2014-2017 by Markus Winand - NO WARRANTY - V20171102
Modifications by Ember Crooks - NO WARRANTY
http://use-the-index-luke.com/s/last_explained

Diagnostic Log and History



Sometimes backups are silently failing

Verify actual backup schedule meets stated backup schedule

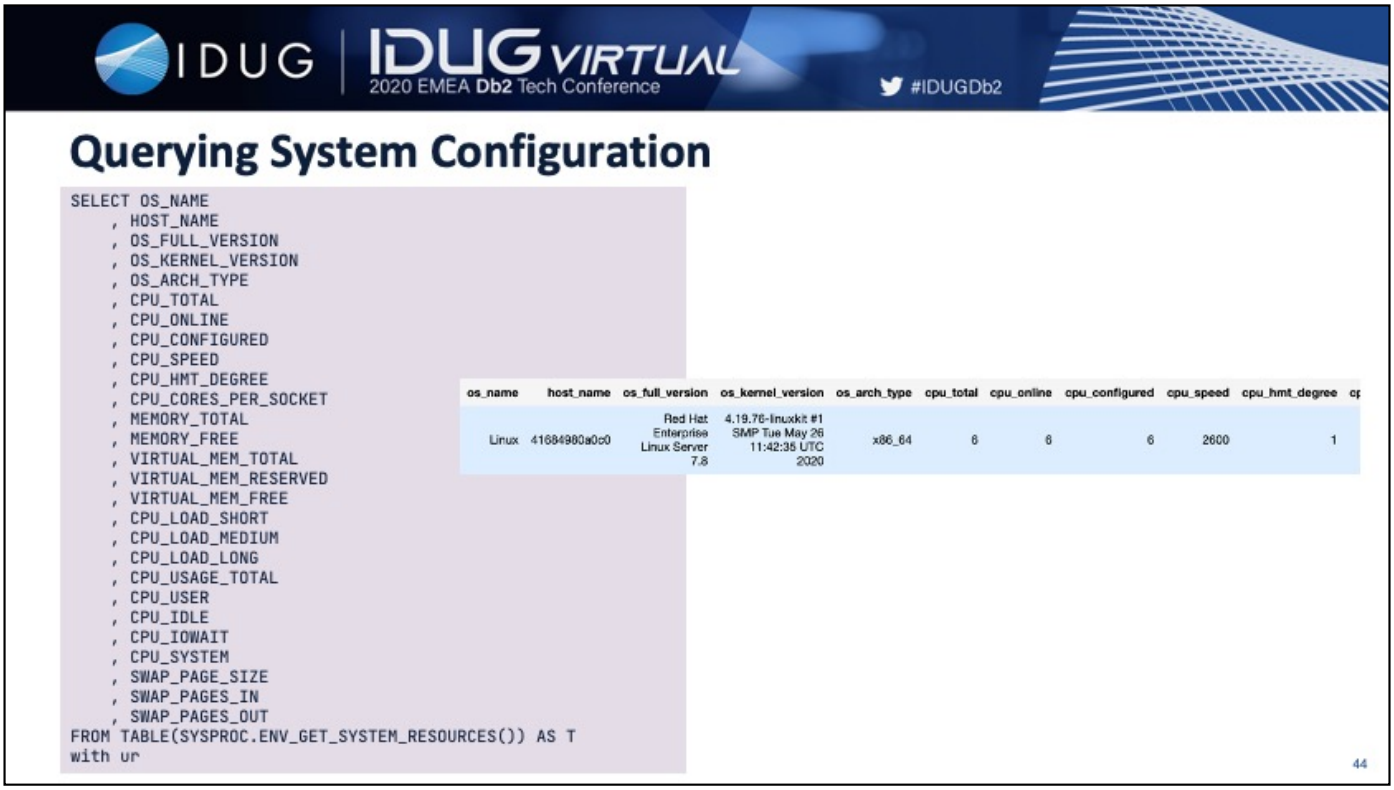
Query the Db2 Diagnostic Log

```
SELECT TIMESTAMP
    , substr(APPL_ID,1,15) as APPL_ID_TRUNC
    , MSGSEVERITY as SEV
    , MSGNUM
    , substr(MSG,1,50) as MSG_trunc
FROM TABLE ( PD_GET_LOG_MSGS( CURRENT_TIMESTAMP - 7 DAYS)) AS T
ORDER BY TIMESTAMP DESC
```

TIMESTAMP	appl_id_trunc	sev	msgnum	msg_trunc
2020-09-07 17:57:52.271192	*LOCAL.db2inst1	W	9502	ADM9502W Index reorganization is complete for tab
2020-09-07 17:57:49.578027	*LOCAL.db2inst1	W	9503	ADM9503W Reorganizing index IID "1" (OBJECTID "56
2020-09-07 17:57:49.559259	*LOCAL.db2inst1	W	9501	ADM9501W Index reorganization has started for tab
2020-09-07 17:57:36.495179	*LOCAL.db2inst1	W	9504	ADM9504W Index reorganization on table *WSCOMUSR.
2020-09-07 17:57:05.077433	*LOCAL.db2inst1	W	9503	ADM9503W Reorganizing index IID "2" (OBJECTID "66
2020-09-07 17:56:42.952230	none	I	1846	ADM1846I Completed archive for log file "S0994447

Use SQL to dig into the details of how your system and database is configured

- System Information
- Db2 Version
- Db2 Licenses
- Db2 Registry
- DBM Configuration
- DB Configuration
- Filesystems/Directories Used by Db2



Warning: Shows the host configuration on docker

Querying Db2 Version

```
SELECT INST_NAME
    , IS_INST_PARTITIONABLE
    , NUM_DBPARTITIONS
    , INST_PTR_SIZE
    , RELEASE_NUM
    , SERVICE_LEVEL
    , BLD_LEVEL
    , PTF
    , FIXPACK_NUM
    , NUM_MEMBERS
FROM SYSIBMADM.ENV_INST_INFO
with ur
```

	inst_name	is_inst_partitionable	num_dbpartitions	inst_ptr_size	release_num	service_level	bld_level	ptf	fixpack_num	num_members
0	db2inst1	0	1	64	0205010F	DB2 v11.1.4.4	s1902261400	DYN1902261400AMD64	4	1

Querying Licensed Db2 Product(s)

```
SELECT INSTALLED_PROD
    , INSTALLED_PROD_FULLNAME
    , LICENSE_INSTALLED
    , PROD_RELEASE
    , LICENSE_TYPE
from SYSIBMADM.ENV_PROD_INFO
with ur
```

	installed_prod	installed_prod_fullname	license_installed	prod_release	license_type
0	ESE	DB2_ENTERPRISE_SERVER_EDITION	Y	11.1	RESTRICTED
1	AESE	DB2_ADVANCED_ENTERPRISE_SERVER_EDITION	N	11.1	None
2	AWSE	DB2_ADVANCED_WORKGROUP_SERVER_EDITION	N	11.1	None
3	WSE	DB2_WORKGROUP_SERVER_EDITION	N	11.1	None
4	DAE	DB2_DIRECT_ADVANCED_EDITION	N	11.1	None
5	DSE	DB2_DIRECT_STANDARD_EDITION	N	11.1	None
6	DEC	DB2_DEVELOPER_C_EDITION	N	11.1	None

Querying Db2 Registry

```
SELECT DBPARTITIONNUM
    , REG_VAR_NAME
    , REG_VAR_VALUE
    , IS_AGGREGATE
    , AGGREGATE_NAME
    , LEVEL
from SYSIBMADM.REG_VARIABLES
order by DBPARTITIONNUM, REG_VAR_NAME
with ur
```

DBPARTITIONNUM	reg_var_name	reg_var_value	is_aggregate	aggregate_name	level
0	DB2BIDI	ON	0	None	I
	DB2COMM	TCPIP	0	None	I
	DB2SYSTEM	[redacted]	0	None	G
	DB2_ANTIJOIN	EXTEND	0	DB2_WORKLOAD	I
	DB2_EVALUNCOMMITTED	YES	0	DB2_WORKLOAD	I
	DB2_HADR_ROS_AVOID_REPLAY_ONLY_WINDOW	ON	0	None	I
	DB2_INLIST_TO_NLJN	YES	0	DB2_WORKLOAD	I

Querying Db2 DBM Configuration

```
SELECT NAME
    , VALUE
    , VALUE_FLAGS
    , DEFERRED_VALUE
    , DEFERRED_VALUE_FLAGS
from SYSIBMADM.DBMCFG
with ur
```

name	VALUE	value_flags	deferred_value	deferred_value_flags
agent_stack_sz	1024	NONE	1024	NONE
agentpri	-1	NONE	-1	NONE
aslheapsz	15	NONE	15	NONE
audit_buf_sz	0	NONE	0	NONE
authentication	SERVER	NONE	SERVER	NONE
catalog_noauth	NO	NONE	NO	NONE
clnt_krb_plugin	None	NONE	None	NONE
clnt_pw_plugin	None	NONE	None	NONE
cluster_mgr	None	NONE	None	NONE

Querying Db2 DB Configuration

```
SELECT NAME
        , VALUE
        , VALUE_FLAGS
        , DEFERRED_VALUE
        , DEFERRED_VALUE_FLAGS
    from SYSIBMADM.DBCFG
    with ur
```

name	VALUE	value_flags	deferred_value	def
app_ctl_heap_sz	256	NONE	256	
appgroup_mem_sz	20000	NONE	20000	
applheapsz	256	AUTOMATIC	256	
archretrydelay	20	NONE	20	
auto_del_rec_obj	OFF	NONE	OFF	
auto_maint	ON	NONE	ON	
auto_db_backup	OFF	NONE	OFF	
auto_tbl_maint	ON	NONE	ON	

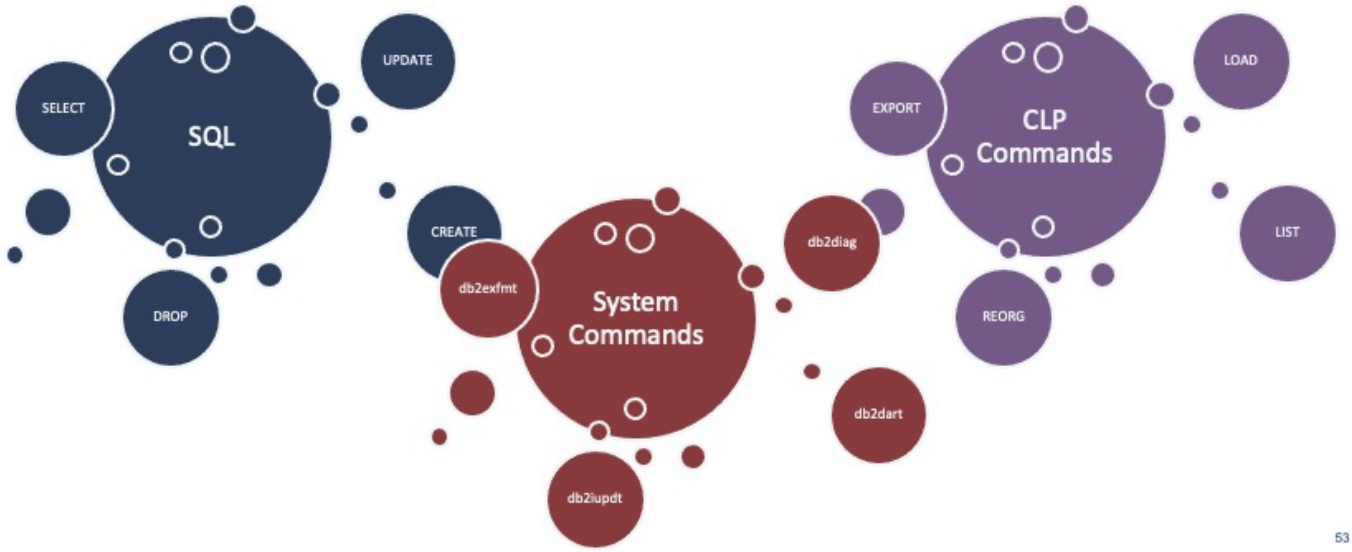
Filesystems or Directories Db2 is Using

```
select *
from sysibmadm.dbpaths
```

DBPARTITIONNUM	TYPE	PATH
0	LOGPATH	[redacted]/db_logs/NODE0000/LOGSTREAM0000/
0	DB_STORAGE_PATH	[redacted]/db_data/
0	DB_STORAGE_PATH	[redacted]/db_data/
0	LOCAL_DB_DIRECTORY	[redacted]/db_data/db2inst1/NODE0000/sqldbdir/
0	DBPATH	[redacted]/db_data/db2inst1/NODE0000/SQL00001/
0	DBPATH	[redacted]/db_data/db2inst1/NODE0000/SQL00001/MEMBER0000/

Beyond Basic SQL – Administrative Commands

SQL vs. Administrative Commands



ADMIN_CMD Procedure

- SQL interfaces and drivers only support SQL, not administrative commands
- Slightly different syntax for the commands may be required
  - Each command has a separate syntax page when used through ADMIN_CMD
- Not all options supported for all commands

Commands Supported for ADMIN_CMD (11.5)

- ADD CONTACT
- ADD CONTACTGROUP
- AUTOCONFIGURE
- BACKUP - online only
- DESCRIBE
- DROP CONTACT
- DROP CONTACTGROUP
- EXPORT
- FORCE APPLICATION
- IMPORT
- INITIALIZE TAPE
- LOAD
- PRUNE HISTORY/LOGFILE
- QUIESCE DATABASE
- QUIESCE TABLESPACES FOR TABLE
- REDISTRIBUTE
- REORG INDEXES/TABLE
- RESET ALERT CONFIGURATION
- RESET DATABASE CONFIGURATION
- RESET DATABASE MANAGER CONFIGURATION
- REWIND TAPE
- RUNSTATS
- SET TAPE POSITION
- UNQUIESCE DATABASE
- UPDATE ALERT CONFIGURATION
- UPDATE CONTACT
- UPDATE CONTACTGROUP
- UPDATE DATABASE CONFIGURATION
- UPDATE DATABASE MANAGER CONFIGURATION
- UPDATE HEALTH NOTIFICATION CONTACT LIST
- UPDATE HISTORY

55

Administrative Routines

- ADMIN_CMD
- ADMIN_COPY_SCHEMA
- ADMIN_DROP_SCHEMA
- ADMIN_EST_INLINE_LENGTH
- ADMIN_GET_ENCRYPTION_INFO
- ADMIN_GET_INDEX_COMPRESS_INFO
- ADMIN_GET_INDEX_INFO
- ADMIN_GET_INTRA_PARALLEL
- ADMIN_GET_MEM_USAGE
- ADMIN_GET_MSGS
- ADMIN_GET_STORAGE_PATHS
- ADMIN_GET_TAB_COMPRESS_INFO
- ADMIN_GET_TAB_DICTIONARY_INFO
- ADMINTABINFO / ADMIN_GET_TAB_INFO
- ADMINTEMPCOLUMNS / ADMIN_GET_TEMP_COLUMNS
- ADMINTEMPTABLES / ADMIN_GET_TEMP_TABLES
- ADMIN_IS_INLINED
- ADMIN_MOVE_TABLE
- ADMIN_MOVE_TABLE_UTIL
- ADMIN_REMOVE_MSGS
- ADMIN_REVALIDATE_DB_OBJECTS
- ADMIN_SET_INTRA_PARALLEL

Using ADMIN_MOVE_TABLE from Jupyter Notebook

```
now_string = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
display(now_string)
```

```
'2020-08-06 12:09:26'
```

```
%sql call ADMIN_MOVE_TABLE([redacted],'BOSEVENT','BOSEVENT_32K','BOSEVENT_32K','BOSEVENT_32K','','','','','','INIT'
```

```
 * db2+ibm_db:/[redacted]
Done.
(ibm_db_dbi.ProgrammingError) ibm_db_dbi::ProgrammingError: The last call to execute did not produce any result set.
(Background on this error at: http://sqlalche.me/e/f405)
```

```
now_string = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
display(now_string)
```

```
'2020-08-06 12:09:37'
```

```
%sql call ADMIN_MOVE_TABLE([redacted],'BOSEVENT','BOSEVENT_32K','BOSEVENT_32K','BOSEVENT_32K','','','','','','COPY'
```

```
 * db2+ibm_db:/[redacted]
Done.
(ibm_db_dbi.ProgrammingError) ibm_db_dbi::ProgrammingError: The last call to execute did not produce any result set.
(Background on this error at: http://sqlalche.me/e/f405)
```

```
now_string = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
display(now_string)
```

```
'2020-08-06 12:16:08'
```

```
%sql call ADMIN_MOVE_TABLE([redacted],'BOSEVENT','BOSEVENT_32K','BOSEVENT_32K','BOSEVENT_32K','','','','','','REPLA
```

```
 * db2+ibm_db:/[redacted]
Done.
(ibm_db_dbi.ProgrammingError) ibm_db_dbi::ProgrammingError: The last call to execute did not produce any result set.
(Background on this error at: http://sqlalche.me/e/f405)
```

```
now_string = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
display(now_string)
```

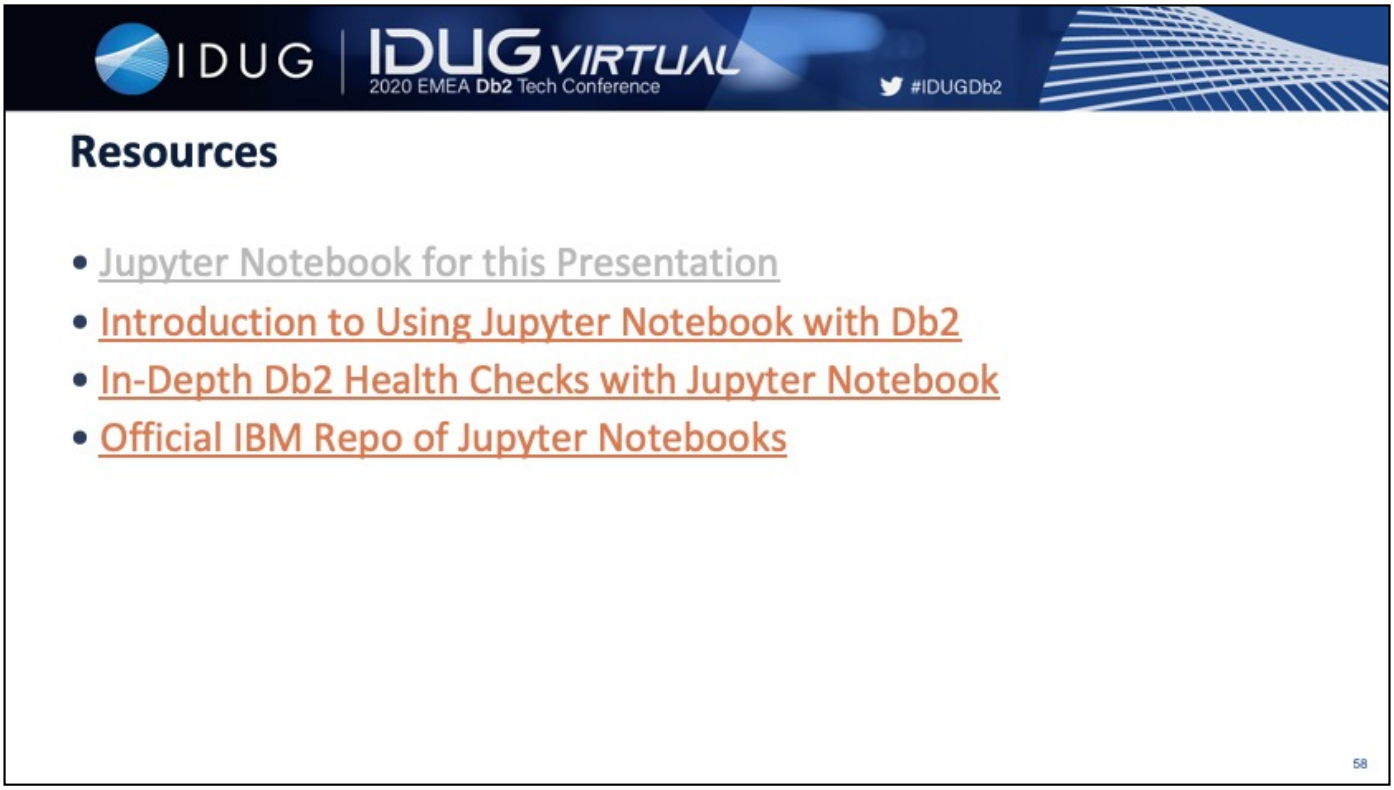
```
'2020-08-06 17:58:03'
```

```
%sql call ADMIN_MOVE_TABLE([redacted],'BOSEVENT','BOSEVENT_32K','BOSEVENT_32K','BOSEVENT_32K','','','','','','SWAP'
```

```
 * db2+ibm_db:/[redacted]
Done.
(ibm_db_dbi.ProgrammingError) ibm_db_dbi::ProgrammingError: The last call to execute did not produce any result set.
(Background on this error at: http://sqlalche.me/e/f405)
```

```
now_string = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
display(now_string)
```

```
'2020-08-06 17:58:37'
```



Jupyter Notebook for this Presentation: https://github.com/ecrooks/db2_and_jupyter_notebooks/blob/master/Db2SQLEverything.ipynb
Introduction to Using Jupyter Notebook with Db2: https://www.dbisoftware.com/blog/db2nightshow.php?id=741
In-Depth Db2 Health Checks with Jupyter Notebook: https://www.dbisoftware.com/blog/db2nightshow.php?id=790
Official IBM Repo of Jupyter Notebooks: https://github.com/IBM/db2-jupyter

Ember has 19+ years of experience with Db2 on Linux, Unix, and Windows platforms. She is the founder and principal author of the popular datageek.blog technical blog where she educates herself and others through example and case study. Ember is an IBM Gold Consultant and IBM Champion in Information Management.