Postgres: Sql, DBA scripts

Links, Tutorials:

http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Appendix.PostgreSQL.CommonDBATasks.html
http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_PostgreSQL.html#d0e99805
http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_BestPractices.html#CHAP_BestPractices.PostgreSQL * * *
https://www.postgresql.org/docs/current/static/routine-vacuuming.html#AUTOVACUUM
http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_LogAccess.Concepts.PostgreSQL.html
http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Storage.html * * *
http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/PostgreSQL.Procedural.Importing.html * * * *
http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Limits.html
https://www.postgresql.org/docs/9.1/static/functions-info.html * * *

Index:
https://www.compose.com/articles/indexing-for-full-text-search-in-postgresql/
http://rachbelaid.com/postgres-full-text-search-is-good-enough/
https://blog.lateral.io/2015/05/full-text-search-in-milliseconds-with-postgresql/
https://blog.codeship.com/unleash-the-power-of-storing-json-in-postgres/

Table Inheritence:
http://stackoverflow.com/questions/3074535/when-to-use-inherited-tables-in-postgresql

Queue Depth:
http://searchsolidstatestorage.techtarget.com/definition/queue-depth

Authentication:

http://stackoverflow.com/questions/4328679/how-to-configure-postgresql-so-it-accepts-loginpassword-auth
http://www.postgresql.org/docs/9.1/static/auth-methods.html

Sql Tricks:

http://postgres.cz/wiki/PostgreSQL_SQL_Tricks

Psql commands:

Connect to Database:
$ psql -d DB_NAME -h localhost -U USER_NAME

Import data into table from a TSV file:
psql> COPY area FROM '/Users/ahsanul.hadi/Documents/Work/db/file_dir' WITH DELIMITER E'\t';
psql> COPY area FROM '/Users/ahsanul.hadi/Documents/Work/db/file_dir' WITH DELIMITER E'\t' NULL AS ;

Pass a sql command:
$ psql -d DB_NAME -h localhost -U USER_NAME -t -A -c "select count(1) from musicbrainz.area";

Run sql from a File. -W will prompt for a password.
$ psql -d DB_NAME -h localhost -U User_name -W -f /Users/ahsanul.hadi/Documents/Work/Database/my_queries.sql

Drop Database:
dropdb -h DB_HOST_NAME -U User_name -e -i <database_name>

Commands:
$ psql -l # List all databases and exits.
$ psql --version # List the version

psql> \c -- DBNAME - USER - HOST -PORT # connects to another database.

\l+ # List of all databases.
\d # List of all tables, views .
\d+ <Table name>q # Describe Table.
SHOW search_path; # show current search_path or \g
SET SET search_path to "__schema_name__"; # Set new search_path.

\dt+ <Table name> # Show table details like: size, owner, type, comment
\d+ <table name> # Shows all column name, storage and column comments.
\dt   # List of all tables.
\dn # List of Schemas
\dv # List of Views.

\di   # List of all Indexes.
\dg # List of Roles. / Users OR \du
\dp # List of Access Privileges.
\db # List of Tablespaces
\dz   # List of installed extensions.
\ds   # List of Sequences.
\df # List of Functions.

\z   # Access privilege
\a   # Align output.
\f   # Field separator
\h   # Help

psql=> \connect nettwerk   # connect to another database.
psql=> \conninfo   # Show current connection info. or \c
psql=> select version();   # Show version info.
psql=> select current_database();   # Show currently connected DB.

DBA scripts:

List of databases:

Get list of Databases

```
SELECT *
FROM  pg_database
WHERE datistemplate is false;
```

Logout/Terminate All other DB connections:

```
SELECT *
FROM pg_stat_activity
ORDER BY query_start DESC;  -- All sessions.

SELECT pg_terminate_backend(pid)
FROM  pg_stat_activity
WHERE pid <> pg_backend_pid();   -- Kill other sessions.
-- WHERE usename = '__user_name__' AND query LIKE '%__query_pattern__%';
--- Kill specific session
```

Get list of Dependant Objects (Without list of columns but with aggregated number of columns that are used):

```
SELECT dependent_ns.nspname  AS dependent_schema,
       dependent_view.relname  AS dependent_view,
       source_ns.nspname   AS source_schema,
       source_table.relname  AS source_table,
       COUNT(pg_attribute.attname) AS Num_of_dependent_cols
       FROM pg_depend
  JOIN pg_rewrite                   ON pg_depend.objid = pg_rewrite.oid
  JOIN pg_class AS dependent_view   ON pg_rewrite.ev_class =
dependent_view.oid
  JOIN pg_class AS source_table     ON pg_depend.refobjid =
source_table.oid
  JOIN pg_attribute                 ON (pg_depend.refobjid =
pg_attribute.attrelid  AND pg_depend.refobjsubid = pg_attribute.attnum)
  JOIN pg_namespace dependent_ns    ON dependent_ns.oid =
dependent_view.relnamespace
  JOIN pg_namespace source_ns       ON source_ns.oid =
source_table.relnamespace
WHERE source_ns.nspname = 'sor_cc_pi'
AND   dependent_ns.nspname <> 'iagdev' /* Exclude this Test Schema */
AND   source_table.relname = 'abx_di_managedarea'
AND   pg_attribute.attnum > 0
--AND pg_attribute.attname = 'my_column'
GROUP BY dependent_ns.nspname,
         dependent_view.relname,
         source_ns.nspname,
         source_table.relname
ORDER BY 1,
         2;
```

Get list of Dependant Objects (With list of columns):

```
SELECT dependent_ns.nspname  AS dependent_schema,
       dependent_view.relname  AS dependent_view,
       source_ns.nspname   AS source_schema,
       source_table.relname  AS source_table,
       pg_attribute.attname  AS column_name
FROM pg_depend
  JOIN pg_rewrite      ON pg_depend.objid = pg_rewrite.oid
  JOIN pg_class AS dependent_view  ON pg_rewrite.ev_class =
dependent_view.oid
  JOIN pg_class AS source_table  ON pg_depend.refobjid =
source_table.oid
  JOIN pg_attribute ON (pg_depend.refobjid = pg_attribute.attrelid AND
pg_depend.refobjsubid = pg_attribute.attnum)
  JOIN pg_namespace dependent_ns  ON dependent_ns.oid =
dependent_view.relnamespace
  JOIN pg_namespace source_ns   ON source_ns.oid =
source_table.relnamespace
WHERE source_ns.nspname = 'sor_cc_pi'
AND   source_table.relname = 'abx_di_managedarea'
AND   pg_attribute.attnum > 0
--AND pg_attribute.attname = 'my_column'
ORDER BY 1,
         2;
```

Related links:

https://www.postgresql.org/docs/9.1/static/catalog-pg-depend.html

https://wiki.postgresql.org/wiki/Pg_depend_display

Get Object Size:

```
SELECT pg_database_size('__db_name__');
SELECT pg_size_pretty(pg_database_size('__db_name__'));

-- Find size of all Databases.
SELECT datname, pg_size_pretty(pg_database_size(datname)) as size
FROM pg_database;


-- Find size of tables and indexes:
SELECT pg_size_pretty(pg_relation_size('public.table_name')); -- This
value exclude indexes and some auxiliary data.
SELECT pg_size_pretty(pg_total_relation_size(‘users’)); -- If you want
to include them use pg_total_relation_size.
SELECT relname, relpages FROM pg_class ORDER BY relpages DESC limit 1;
-- find the largest table in the postgreSQL database.
```

List of all Constraints and related info:

```
-- Check > information_schema.constraint_column_usage
-- Check > information_schema.key_column_usage

SELECT *
FROM information_schema.table_constraints
WHERE table_schema = '__schema_name__'
AND constraint_type = 'CHECK';
```

List all tables and indexes:

```
--  Create a View for this.
SELECT schemaname AS schema_name,
       relname AS table_name,
       pg_size_pretty(pg_relation_size (schemaname || '.' || relname))
AS size
FROM (SELECT schemaname,
             relname,
             'table' AS TYPE
      FROM pg_stat_user_tables
      UNION ALL
      SELECT schemaname,
             relname,
             'index' AS TYPE
      FROM pg_stat_user_indexes) x;
```

Get all Comments (Table, Columns):

```
SELECT a.table_name,
       a.objsubid,
       b.column_name,
       a.description
FROM (SELECT nspname AS table_schema,
             relname AS table_name,
             objsubid,
             description
      FROM pg_description
        JOIN pg_class ON pg_description.objoid = pg_class.oid
        JOIN pg_namespace ON pg_class.relnamespace = pg_namespace.oid
      WHERE nspname = '__schema_name__') AS a
  LEFT JOIN (SELECT c.table_schema,
                    c.table_name,
                    c.ordinal_position,
                    c.column_name,
                    pgd.description
             FROM pg_catalog.pg_statio_all_tables AS st
               INNER JOIN pg_catalog.pg_description AS pgd ON
(pgd.objoid = st.relid)
               INNER JOIN information_schema.columns AS c
                       ON (pgd.objsubid = c.ordinal_position
                      AND c.table_schema = st.schemaname
                      AND c.table_name = st.relname)
             WHERE c.table_schema = '__schema_name__') AS b
         ON a.table_name = b.table_name
        AND a.objsubid = b.ordinal_position
ORDER BY a.table_name,
         a.objsubid;
```

Find out whether a table is too fragmented (Table Bloat) or not:

```
-- Check how many OS pages are allocated to the table. (works on
Greenplum)

ANALYZE <table-name>;

SELECT relname,relpages,reltuples
  FROM pg_class
 WHERE relname = <table-name>;

-- Try running a vacuum full on the table and see if that make a
difference.

VACUUM <table-name>;
REINDEX <table-name>;
ANALYZE <table-name>;
```

PGCLI:

-- for Ubuntu.

-- http://pgcli.com/install

```
$ which pip  # check whether exists or not ?
$ sudo su  # switch to Root user.
$ apt-get update # update package list.
$ apt-get install python-pip
$ apt-get install libpq-dev python-dev
$ pip install pgcli
```

Use Regexp (Regular Expression):

https://www.postgresql.org/docs/9.3/static/functions-matching.html

http://blog.lerner.co.il/regexps-in-postgresql/

-- example.

```
SELECT col_name,
       REGEXP_REPLACE(col_name,E '[\\n\\r\\u2028]+',' ','g') AS
col_name_new, --- replace 'new lines'
       counts,
       (SELECT COUNT(*) FROM REGEXP_MATCHES(col_name,'\|','g')) AS
num_of_bar-- then we need to find the 'BAR' separated values.
       FROM (SELECT UPPER(TRIM(col_name)) AS col_name,
                    COUNT(1) AS counts
             FROM public.table1
             WHERE col_name IS NOT NULL
             GROUP BY UPPER(TRIM(col_name))) AS t
ORDER BY col_name;
```

Backup and restore:

```
-- Take FULL BACKUP of source database.
$ pg_dump -h [host_name] -d [database_name] -U [user_name] -n
[schema_name] -C -c --if-exists --no-privileges --no-owner
--no-tablespaces > /Users/backup/Full_data_dump.sql

-- Drop the schema from the target database
$ psql -h [host_name] -d [database_name]  -U [user_name] -c 'DROP SCHEMA
[schema_name] CASCADE';

-- IMPORT the schema data in your target database.
$ PGOPTIONS='--client-min-messages=warning' psql -h [host_name] -d
[database_name]  -U [user_name] -v ON_ERROR_STOP=ON -X -q
--single-transaction -f /Users/backup/Full_data_dump.sql



-- Export in CSV format
SELECT '\COPY ' || tablename || ' TO ''/Users/backup/' || tablename ||
'.csv'' WITH (FORMAT CSV, HEADER);' as cmd
FROM pg_tables
WHERE schemaname = 'table_name'
ORDER BY tablename asc;


-- EXPORT Data for a schema and excluding the Metadata tables. Also
disable triggers.
pg_dump -h [host_name] -d [database_name] -U [user_name] -n
[schema_name] -T [table_name] -T [table_name] -a --disable-triggers
--no-privileges --no-owner > /Users/backup/data_only_export.sql

-- EXPORT Database structure for a schema Only.
pg_dump -h [host_name] -d [database_name] -U [user_name] -n
[schema_name] -C -c -s --if-exists  --no-privileges --no-owner
--no-tablespaces > /Users/backup/schema_only_export.sql
```

Parallel pg_dump job:

```
$ time pg_dump -h [host_name] -d [database_name] -U [user_name] -w -n
[schema_name] -O -Fd -j 6 -f /Users/backup
```