Task 6

Difference between Oracle and Postgres

- Licensing: Oracle is a proprietary software and requires a license for use, while PostgreSQL is open source and free to use.
- Platform Support: Oracle supports more platforms than PostgreSQL, including Solaris, HP-UX, and IBM AIX. However, PostgreSQL supports more operating systems, including Linux, macOS, Windows, and FreeBSD.
- Scalability: Both PostgreSQL and Oracle are highly scalable, but Oracle has a slight edge in scalability due to its more advanced clustering features.
- Performance: In general, Oracle performs better than PostgreSQL in large-scale enterprise environments, but PostgreSQL is known for its efficient use of resources and good performance in smaller applications.
- Data Types: PostgreSQL has a wider range of data types than Oracle, including arrays, hstore (key-value pairs), and JSON data types.
- SQL Compatibility: Both PostgreSQL and Oracle support SQL, but they have some differences in syntax and implementation. For example, PostgreSQL supports more SQL standards than Oracle.
- Replication: Both PostgreSQL and Oracle support replication, but PostgreSQL's replication features are easier to set up and manage.
- Backup and Recovery: Oracle has more advanced backup and recovery features, including point-in-time recovery and flashback database. PostgreSQL has similar features, but they are not as advanced as Oracle's.
- Development Community: PostgreSQL has a large and active open source community, which means that there are many resources available for developers. Oracle has a smaller community, but it is still very active.
- Built-in Functions: Both PostgreSQL and Oracle provide a wide range of built-in functions for data manipulation and analysis. However, PostgreSQL has more advanced data types and functions than Oracle, including support for arrays, hstore, and JSON data types.
- Queries: Both PostgreSQL and Oracle support SQL queries and have similar syntax. However, PostgreSQL supports more SQL standards than Oracle, which means that it has more advanced query features.

PostgreSQL also supports window functions, which are not available in Oracle.

- PL/SQL: PL/SQL is Oracle's procedural language for SQL, while PostgreSQL supports PL/pgSQL. Both languages are similar in syntax and functionality, but there are some differences in their features. For example, PL/SQL has more built-in functions than PL/pgSQL, but PL/pgSQL has better support for arrays and hstore.
- Stored Procedures: Both PostgreSQL and Oracle support stored procedures, which are user-defined routines that can be called from SQL queries. However, Oracle's stored procedures can be more complex and powerful than PostgreSQL's, thanks to its advanced PL/SQL language.
- Triggers: Both PostgreSQL and Oracle support triggers, which are stored procedures that are automatically executed in response to certain events. However, Oracle's triggers are more powerful and flexible than PostgreSQL's, thanks to its advanced PL/SQL language.
- Views: Both PostgreSQL and Oracle support views, which are virtual tables that are defined by SQL queries. However, Oracle's views can be more complex and powerful than PostgreSQL's, thanks to its advanced query features and PL/SQL language.
- One major difference between Oracle and PostgreSQL is their approach to transaction handling. In Oracle, transactions are managed using the Automatic Undo Management (AUM) feature, which uses a dedicated tablespace for storing undo data. This allows Oracle to support very large transactions and provides a high level of transaction concurrency. In contrast, PostgreSQL uses a multi-version concurrency control (MVCC) approach to transaction handling. MVCC creates a new version of a row in the database when it is updated, rather than overwriting the existing row. This approach allows PostgreSQL to support a high level of concurrency and provides a consistent view of the database, even when multiple transactions are occurring simultaneously.

  As a result of these different approaches to transaction handling, Oracle and PostgreSQL have different strengths and weaknesses. Oracle is known for its ability to handle very large transactions and its advanced concurrency control features. PostgreSQL is known for its efficient use of resources and its ability to handle many small transactions concurrently. For example, in an e-commerce application where there are many small transactions happening concurrently, PostgreSQL may be a better choice

due to its efficient use of resources and ability to handle many transactions simultaneously. However, in a financial application where there are large transactions happening, Oracle may be a better choice due to its ability to handle very large transactions and advanced concurrency control features.

Difference based on syntax, function and sequence, view

Oracle	Postgres
select sysdate from dual	*select 'now'::datetime* There is no "dual" table Unlike other RDBMS, PostgreSQL allows a "select" without the "from" clause. This use does not affect portability because the syntax to get current time is already DBMS specific.
Oracle relational operators may have a space between the characters. For example, the following select will work: *select id, name from employee where id > = 10;*	PostgreSQL relational operators doesn't allow spaces, the characters that compound an operator must be consecutive when the command is parsed: *select id, name from employee where id >= 10;*
CREATE SEQUENCE seqname [INCREMENT BY integer] [MINVALUE integer] [MAXVALUE integer] [START WITH integer] [CACHE integer] [CYCLE | NOCYCLE] , Oracle's "create sequence" has other arguments not listed here and not supported by PostgreSQL, but the main difference is the need of 'by' and "with" after "increment" and "start". If you don't specify "MAXVALUE" or if you use the parameter	*CREATE SEQUENCE seqname [INCREMENT increment] [MINVALUE minvalue] [MAXVALUE maxvalue] [START start] [CACHE cache] [CYCLE]* If you don't specify MAXVALUE, then the maximum value is 2147483647 for ascending sequences.

"NOMAXVALUE", then the actual limit is 1027	
A view can be "updatable" if some conditions are satisfied.	Views are read only.
Hierarchical queries – "CONNECT BY"	Nothing similar
To return the current value and increment the counter: *sequence_name.nextval;* Possible usage in a select statement: *select sequence_name.nextval from dual;*	To return the current value and increment the counter: *nextval('sequence_name');* Possible usage in a select statement *select nextval('sequence_name');* Note that unlike other RDBMS, PostgreSQL allows a select without the 'from' clause. This use does not affect portability because the sequence syntax is already DBMS specific.
Interactive command prompt tool: SQL*Plus	Interactive command prompt tool: psql
The "ROWNUM" pseudo-column returns a number indicating the order in which Oracle selects the row.	There isn't anything equivalent to Oracle ROWNUM. However, you can limit the number of rows returned by a query using the "LIMIT" clause:

ROWNUM can be used to limit the number of rows returned by a query, for example: *select * from employees where rownum < 10 order by name;* ROWNUM can be used in the projection as one of the values returned by the query (first line has value 1, second line value 2, and so on): *select rownum, name from employees order by name;*	*select * from employees order by name limit 10;* In some cases, it's possible that the pseudo-column OID may substitute ROWNUM, although they have different behavior. OID is a unique identifier of each line per table, while ROWNUM is always 1, 2, …, N for each different query. *select oid, name from employees order by name;* And the query that uses ROWNUM can have join tables. If your select is a join you'll have a different OID for each table, because each one has an OID column.
select unique col1, col2 from table1 In Oracle "distinct" and "unique" keywords are synonymous in the select statement.	*select distinct col1, col2 from table1* PostgreSQL doesn't allow "select unique".
SELECT product_id FROM inventories *MINUS* *SELECT product_id FROM order_items;*	*SELECT product_id FROM inventories* *EXCEPT* *SELECT product_id FROM order_items;*